2000

Algorithms for enhancing pattern separability, feature selection and incremental learning with applications to gas sensing electronic nose systems

Robi Polikar
Iowa State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Algorithms for enhancing pattern separability, feature selection and incremental learning with applications to gas sensing electronic nose systems

by

Robi Polikar

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Co-majors: Electrical Engineering (Communications and Signal Processing);

Biomedical Engineering

Major Professors: Lalita Udpa and Mary Helen Greer

Iowa State University

Ames, Iowa

2000

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Robi Polikar

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Co-major Professor

Signature was redacted for privacy.

Co-major Professor

Signature was redacted for privacy.

For the Co-major Program

Signature was redacted for privacy.

For the Co-major Program

Signature was redacted for privacy.

For the Graduate College

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

A Ph.D. dissertation is never the sole work of one person, and this dissertation is no exception. This work would have never been completed without the enthusiastic support, help and guidance of many people, to whom I am gratefully indebted.

First and foremost, I would like to express my greatest gratitude to my major professors, Dr. Lalita Udpa and Dr. Mary Helen Greer. They have been most influential in my achievements during my graduate studies, as they helped me to grow professionally and become a better researcher and a better teacher. I consider myself very fortunate and privileged to have them as my major professors, since without their academic guidance, personal and financial support, as well as friendship, my past seven years here at Iowa State would not have been such a pleasant and memorable experience.

I also would like to thank my committee members, Dr. Satish Udpa, Dr. Marc Porter and Dr. Eric Bartlett for their academic guidance and support. In particular, I would like to acknowledge the guidance of Dr. Satish Udpa, who has shown as much interest in my work and my academic education as a Ph.D. student can only hope to see from his/her major professor. I would also like to acknowledge Dr. Porter for introducing me to the magical world of analytical chemistry, since the work described in this dissertation is centered around the gas sensing and electronic nose applications we have investigated in his laboratory. However, my attempts in understanding the chemical concepts described in this work would have all been in vain without the invaluable help of Dr. Ruth Shinar of Microanalytical Instrumentation Center in Ames Lab. She has patiently sat with me and spent countless hours to provide me with all the analytical chemistry background I needed. Although not formally involved in my

graduate committee, she has shown genuine interest in my work, and helped in many ways that I couldn't even expect from a committee member. I also would like to acknowledge Dr. Vasant Honavar, whose class has been an inspiration for my work in incremental learning.

Every time a graduate student feels as if s/he is stuck with a particular problem, the first help typically comes from fellow graduate students. This has certainly been the case for me, and I would like to acknowledge Pradeep Ramuhalli for many fruitful discussions, for our collaboration in many projects, demos, and trips, as well as for his true friendship.

Often forgotten in acknowledgements are departmental secretaries and other staff, without whose help no student can get paid or graduate in time. After all, they are the ones who do all the paper work, remind all the deadlines, and make sure that everything is in order. Therefore, my heartfelt thanks go to Mrs. Linda Clifford (associate God), Pam Myers, and Nancy Knight for their assistance.

What makes a university more than just a series of lectures, and endless hours of studying is the social and cultural activities it provides to its students. In my case, ISU Dance has provided me with the extracurricular activities that I needed to keep my sanity during my years as a graduate student. I am very grateful to Janice Baker and Laurie Sanda for giving me the opportunity to take active place in various live performing arts events. I feel very privileged to be part of the campus dance organizations, such as Orchesis I, Orchesis II and ISU Ballroom Dance Club, and to perform in our annual dance concerts such as Barjche, FootFalls and Images of Dance.

Most importantly, I would like to thank my parents who have endured countless number of emotional and financial sacrifices and difficulties for me to have the best education possible. Without their love and support, I certainly would not be where I am today.

ABSTRACT

Three major issues in pattern recognition and data analysis have been addressed in this study and applied to the problem of identification of volatile organic compounds (VOC) for gas sensing applications. Various approaches have been proposed and discussed. These approaches are not only applicable to the VOC identification, but also to a variety of pattern recognition and data analysis problems. In particular,

- enhancing pattern separability for challenging classification problems,
- optimum feature selection problem, and
- incremental learning for neural networks

have been investigated.

Three different approaches are proposed for enhancing pattern separability for classification of closely spaced, or possibly overlapping clusters. In the neurofuzzy approach, a fuzzy inference system that considers the dynamic ranges of individual features is developed. Feature range stretching (FRS) is introduced as an alternative approach for increasing intercluster distances by mapping the tight dynamic range of each feature to a wider range through a nonlinear function. Finally, a third approach, nonlinear cluster transformation (NCT), is proposed, which increases intercluster distances while preserving intracluster distances. It is shown that NCT achieves comparable, or better, performance than the other two methods at a fraction of the computational burden. The implementation issues and relative advantages and disadvantages of these approaches are systematically investigated.

Selection of optimum features is addressed using both a decision tree based approach, and a wrapper approach. The hill-climb search based wrapper approach is applied for selection of the optimum features for gas sensing problems.

Finally, a new method, Learn++, is proposed that gives classification algorithms, the capability of incrementally learning from new data. Learn++ is introduced for incremental learning of new data, when the original database is no longer available. Learn++ algorithm is based on strategically combining an ensemble of classifiers, each of which is trained to learn only a small portion of the pattern space. Furthermore, Learn++ is capable of learning new data even when new classes are introduced, and it also features a built-in mechanism for estimating the reliability of its classification decision.

All proposed methods are explained in detail and simulation results are discussed along with directions for future work

CHAPTER 1

INTRODUCTION

1.1 Emerging Interdisciplinary Problems

As we enter the 21st century, our technological advancements allow us to solve increasingly complicated and challenging problems, which can no longer be solved by professionals with a single area of expertise. As a direct consequence of this, traditional boundaries among different disciplines are disappearing at a remarkable pace. Researchers and professionals in various disciplines now realize that their work, once thought of as independent or unrelated to other fields, can no longer be isolated from other disciplines. This is simply because the challenges we face today are of truly interdisciplinary nature. Such challenges require that researchers and professionals collaborate with their colleagues in other disciplines, since overcoming these challenges requires expert knowledge of various fields. Probably one of the truly notable benefits of this cooperation is that it allows collaborators to learn about each other's fields, which in turn allows rapid proliferation of interdisciplinary problem solving techniques.

This dissertation is a prime example of such collaboration. The problem undertaken is the automated identification of volatile organic compounds (VOCs), which enjoys increasing importance and attention among analytical chemists; however, a closer look at how such a system can be realized, and what it is capable of above and beyond identifying VOCs, reveals the true interdisciplinary nature of the problem.

First, identification of VOCs requires a physically measurable response, which is typically a signal of electrical origin, to be detected and recorded. Successful detection of a signal, however, requires various filtering and denoising schemes, which calls for expertise in signal processing, the first clue that the problem is closely related to electrical engineering. Furthermore, automated identification of signals calls for yet another area of electrical engineering, namely pattern recognition. In addition, novel pattern recognition techniques are results of joint efforts in artificial intelligence and machine learning, which are areas of major research interest in computer science. Finally, successful identification of VOCs can suggest innovative methods for the more general problem of gas sensing. This in turn can guide us in our efforts in modeling and artificially implementing the last human sensory system that is not yet electronically or mechanically implemented: the olfactory system. Hence, automated identification of VOCs, in fact encompasses the fields of computer science, anatomy, physiology, electrical engineering and biomedical engineering, as well as analytical chemistry.

In this dissertation, all aspects of this problem as they relate to the above listed fields are investigated and discussed. The experimental procedures followed for obtaining physically measurable signals from VOCs, the properties of the chemicals used, and how such a VOC identification system relates to mammalian olfactory system are all discussed in the introductory chapters of this dissertation. Pattern recognition schemes developed for enhancing separability of patterns of overlapping clusters, selection of optimum features for successful pattern recognition, and the algorithm developed for incremental learning of new data constitute the electrical engineering, artificial intelligence and computer science aspects of this work.

1.2 Background and Motivation

Identification and quantification of volatile organic compounds (VOCs) are of crucial importance for environmental monitoring of air, soil and groundwater, as well as for many industries and organizations, particularly for those involved in various applications of gas sensing. VOC molecules are detected by piezoelectric quartz crystals, which respond to gas exposure as a shifted resonant frequency. The frequency responses of crystals to various VOCs are then used to identify the VOCs by appropriate signal processing and pattern recognition techniques.

As shown by many researchers, a solution to the problem of detecting and identifying VOCs is also a solution to the problem of detecting and identifying many other gases, whether these gases are emitted from food items, anti-personnel mines, hazardous chemicals, or illegal drugs and plastic bombs. Developing such a system is therefore of great interest to

- food industries for testing the quality or wholesomeness of a particular food product [1, 2, 3] ,
- military and humanitarian organizations for locating buried landmines [4],
- petrochemical industries and gas valve manufacturing companies for detecting and identifying hazardous gases [5, 6, 7, 8, 9],
- airport security and customs inspection for detecting illegal drugs and plastic bombs [10], etc.

As a consequence of the increasing interest in gas sensing over the last 15 years, a growing number of experimental techniques and signal processing algorithms have been developed for improving signal quality, and a number of pattern recognition algorithms have been developed for identification and classification of VOCs, as well as other gases.

The experimental techniques and signal processing algorithms have mainly targeted the enhancement of signal to noise ratio in sensor responses and extracting important features from sensor responses so that they can be accurately identified by using subsequent pattern recognition techniques. Such experimental techniques include preconcentration of gases [6], and precise controlling of VOC flow fluctuations [11], whereas signal processing algorithms include drift compensation [12], improving gas sensor response time and signal separability through transient analysis [13,14]. Various well established signal processing techniques such as Gram – Schmidt orthogonalization, Fourier and wavelet analysis [15] have also been used. Modeling schemes, such as olfactory modeling [15], computational chemistry [16], and electrical circuit equivalents of crystals [17] have been tried for simulating and predicting the responses of the crystals.

Among pattern recognition algorithms, cluster analysis and principal component analysis (PCA) [1, 2, 3, 5, 6, 18, 19, 20, 21], extended disjoint principal components regression [22], k-means, [23] various neural network architectures [19, 21, 24, 25, 26, 27, 28, 29, 30, 31], neurofuzzy approaches [29, 32, 33, 34, 35] and more recently genetic algorithms [36] have achieved significant success for the VOC identification problem. As VOC identification improved, attention was directed to more challenging issues, such as identification of VOCs in mixtures [7, 22, 37, 38, 39], and/or in environments of varying temperature and humidity [40, 41, 42], or under the soil [43]. In particular, identification of VOCs in mixtures has proven to be exceptionally difficult due to competing interactions between the individual components of the mixture and the coating material. Such competition results in patterns that significantly overlap in the feature space, which considerably complicates the classification problem.

Since the performance of even the best pattern recognition algorithm is always limited by the quality of the data it is applied to, it is evident that the coating selection must be made strategically. Intelligent selection of the coatings, tailored towards the chemical and physical properties of the VOCs to be detected, can significantly simplify the identification of the VOCs. Therefore, utmost care must be shown to select the optimal set of coatings for the particular application.

There are two stages for choosing an optimum set of coatings. First, potentially useful coatings with desired chemical and physical properties are determined based on the solubility properties of the VOCs. Although the procedure for doing so is well established in the analytical chemistry community [44, 45, 46], determining the best set of coatings among the potentially useful ones is a daunting task, due to overwhelmingly large combinations of possible coatings. Therefore, an optimization scheme is required to choose the optimum and the smallest subset of the potentially useful coatings [22].

Another important issue in identification of VOCs through pattern recognition that has not been addressed so far is the problem of integrating a new set of data that may later become available. This problem emerges when a previously designed pattern recognition algorithm trained with existing data achieves poor performance with similar data obtained from a different site, or obtained under slightly different conditions. To make the problem even more challenging, it might be necessary to be able to identify additional VOCs which were previously not present.

It is interesting to note that the above mentioned issues are special cases of age-old problems in the areas of signal processing, pattern recognition and cognitive learning. In particular, identification of mixtures of VOCs has been a very challenging problem due to overlap-

ping of signature patterns of VOCs; therefore, it is a special case of the more general problem of enhancing pattern separability for classification. Similarly, selection of optimum coatings is a special case of the optimal feature selection problem, and the ability to incorporate new data into an existing classifier is a special case of incremental learning. All three issues mentioned above are of significant interest to pattern recognition, artificial intelligence, cognitive learning, and neural network communities.

The final cumulative goal of the research in the gas sensing area is developing a sensory device, complete with its hardware and software, to detect, identify and quantify various odors of interest in the environment. Such a device, mimicking the mammalian olfactory system, is affectionately referred to as *the electronic nose*.

This research started out with a modest goal of identifying individual VOCs from their signature patterns. However, it grew and evolved significantly over the last few years, not only to solve the issues related to VOC identification, but also to solve the above listed more general and challenging problems.

1.3 Organization of the Dissertation

Several methods have been developed to address the issues discussed above and the proposed approaches along with results are presented in this dissertation: Chapter 2 presents a comprehensive review of the mammalian olfactory system which any electronic nose system is expected to replicate. In particular, the human olfactory system is presented to set a benchmark for current and future electronic nose systems. A comparison between the anatomical structures used in the mammalian olfactory system and the devices / procedures that are used to replicate these structures are discussed. Individual components of state-of-the-art

electronic nose systems are introduced and described. Finally, commercially available current electronic nose systems are introduced.

Chapter 3 first provides the necessary chemistry background for gas sensing, in particular as it applies to VOC detection using polymer coated quartz crystal microbalances. A review of chemical and physical properties of coatings and those of VOCs that need to be considered for making coating selection is given. Finally, the separability issues and the problem of overlapping clusters regarding this database are also explained.

Techniques for enhancing pattern separability that are developed are discussed in Chapter 4. These techniques are mainly applied to identification of mixtures of VOCs though they are applicable to any pattern recognition problem. Three different methods are proposed, tested and compared, namely, fuzzy processing, feature range stretching, and nonlinear cluster transformation.

Existing schemes for optimal selection of coatings are first reviewed in Chapter 5, followed by two approaches that were developed to reduce the computational complexity of the existing schemes.

The problem of incorporating new information into a classifier is investigated in Chapter 6, in the context of incremental learning. A theoretical upper bound for Learn++'s training error is derived. The performance of Learn++ on incrementally learning new data, which may include new classes, is presented not only for the VOC database, but also for a number of synthetic and real world databases.

Conclusions and discussions are finally given in Chapter 7, along with directions for future research in related areas. The overall structure of this work is illustrated in Figure 1.1. Due to the independent nature of the individual issues addressed in this research, each chap-

ter is further divided into its own subsection of introduction and literature review, method, results, discussion and conclusions.

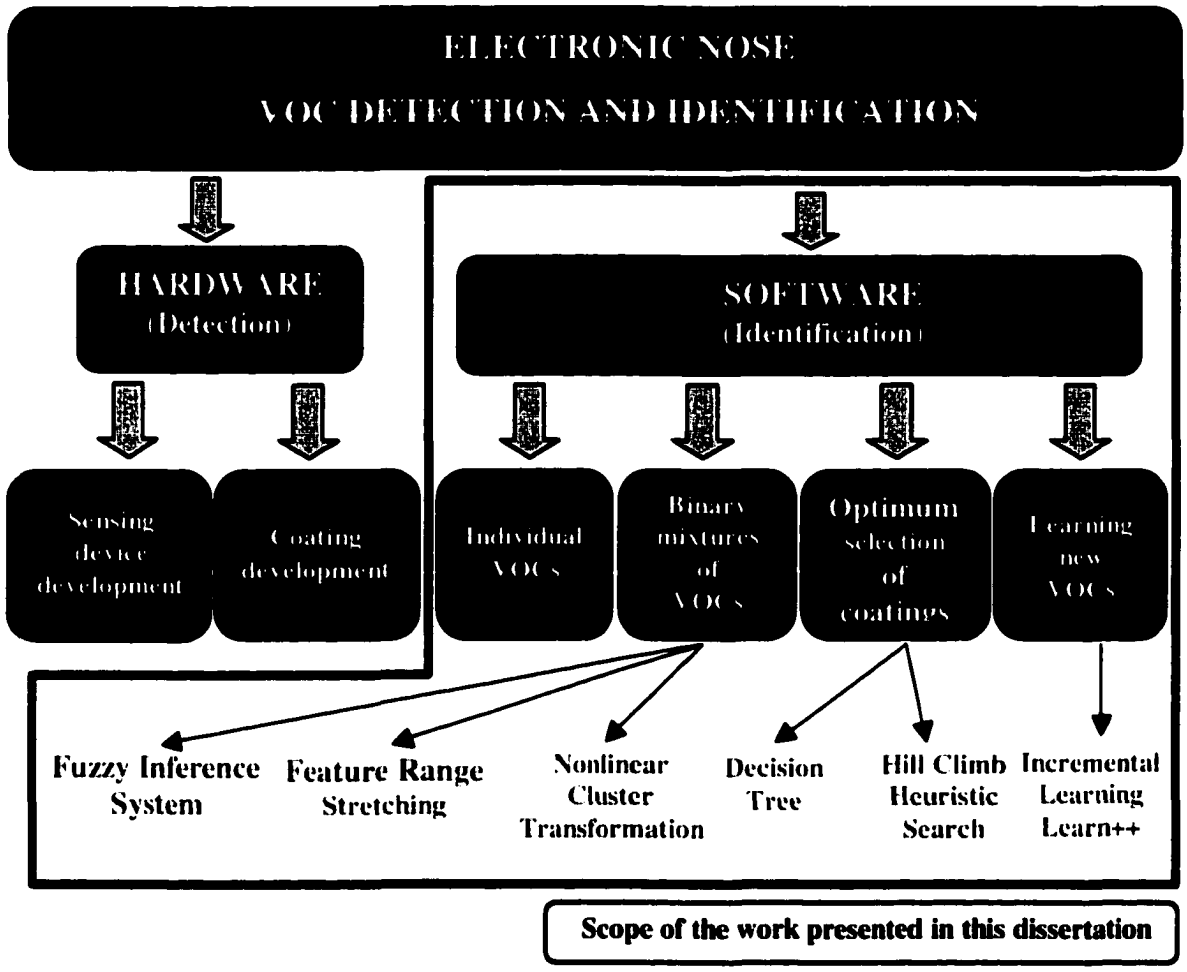


Figure 1.1 Overall structure of the VOC detection and identification problem

CHAPTER 2

THE MAMMALIAN OLFACTORY SYSTEM AND THE QUEST FOR ELECTRONIC NOSE

2.1 Introduction

Many of the topics investigated in this research have arisen from the gas identification problem, the ultimate goal of which is to be able to build an electronic nose for various gas sensing applications. This endeavor of emulating the olfactory system, however, is a daunting task, considering the complexity of the system. In order to appreciate the complexity and the preciseness of this system and to set a performance benchmark, this chapter is mainly devoted to olfactory physiology.

The environment we live in has a wealth of chemical information, and most species are equipped with a sensory system to make the most of this chemical information. This system, known as the olfactory system, provides a very powerful means of chemical communication among various species, particularly among mammals.

Mammals have one of the most advanced olfactory organs among all species, because for many mammals olfaction is the prime way of communication to attract others of the same species or to detect predators and hazardous conditions, as well as for mating, marking territory, etc. Although the human olfactory system is significantly less complicated than those of some other mammals (such as cats, dogs, rabbits, etc.), it is still far more sophisticated than any electronic nose system available today.

Various introductory topics regarding olfactory physiology are reviewed in this chapter. In particular, the anatomy of the human olfactory system is given in Section 2.2. The physiology of olfaction in mammals is then explained in Section 2.3, which includes discussions on olfactory stimulation, the structure of olfactory receptors and the physiological procedure for signal transduction (converting the chemical information to an electrical signal). Olfactory pathways for transmitting the odorant information to be processed in brain are discussed in Section 2.4, whereas the sensitivity and the selectivity of olfactory receptors in resolving various odors are given in Section 2.5. The first five sections of this chapter are meant to be a comprehensive overview of the human olfactory system and have been compiled from various texts including [47, 48, 49, 50, 51, 52, 53]. In Section 2.6, a brief overview of the emerging electronic nose technologies are presented, where individual components of a typical electronic nose system are discussed and compared to those that are present in the olfactory system. A list of commercially available electronic nose systems is also provided in this section.

2.2 The Anatomy of the Olfactory System

Olfaction occurs as a result of the interaction between the odorant molecules and the olfactory receptors, which are located in the superior region of the nasal cavity. It should be noted that most of the nasal cavity is devoted to respiration, with only a small region on the roof of the nasal cavity, called the *olfactory recess*, involved in olfaction. The actual receptors employed in detection of the odor are called *olfactory receptor cells*, which are located in a specialized epithelial layer, called the *olfactory epithelium*. There are approximately 10 ~

20 million olfactory receptor cells within the olfactory epithelium. Each receptor cell is actually a neuron, and therefore olfactory receptors are also called *olfactory neurons* [50].

Olfactory neurons are quite unique among all other neurons in many aspects. First, olfactory neurons are bipolar neurons. Their axons and dendrites extend from opposite sides of the cell body (soma). Bipolar neurons are only found in sensory organs acting as receptors. Second, olfactory neurons are the only neurons that can constantly replace themselves. In fact, the entire olfactory epithelium degenerates and is lost every sixty days, only to be reproduced by *basal cells*. Olfactory receptor neurons, along with basal cells, and *supporting (sustantecular) cells* constitute the three major cell types present in the olfactory system, as shown in Figure 2.1. Olfactory neurons are sandwiched between the cushioning columnar supporting cells, which make up the most of the olfactory epithelium. A yellowish-brown pigment contained in supporting cells gives the characteristic color to the olfactory epithelium.

Each olfactory receptor has an extensive network of dendrites, from which several long hairlike processes called *cilia* radiate. Cilia are roughly 2 μm long with a diameter of 0.1 μm. There are approximately 10 – 20 cilia per neuron, and their main function is to increase the receptive surface area of the olfactory neurons. It should be noted that although the surface area of the olfactory epithelium is about 5 cm^2, the total surface area including the cilia is larger than a typical human body. As a comparison, the surface area of the olfactory epithelium of a dog is 72 times that of a human [50, 53].

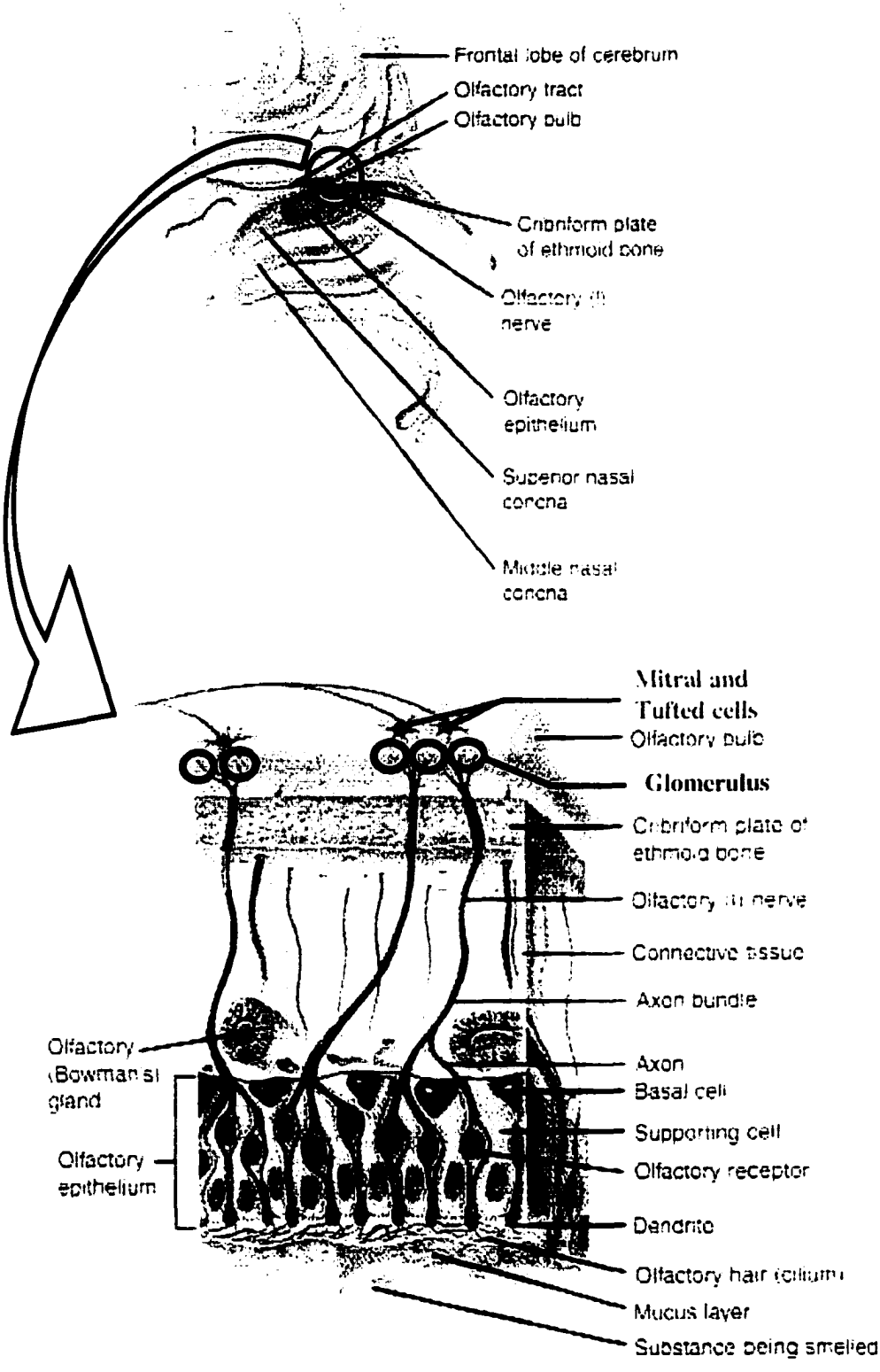


Figure 2.1 The anatomy of the olfactory system [from 48]

Scattered around the axons of the olfactory neurons are the *Bowman's glands*, which produce the mucus layer covering the olfactory epithelium. The mucus, a polysaccharide solution that contains various enzymes, antibodies, salts and special proteins, constantly covers the olfactory epithelium, and it is renewed every ten minutes. The special proteins in the mucus are the odorant binding proteins, which facilitate the interaction of odorant molecules with the olfactory neurons.

Note that the nasal cavity provides an access to the brain, and therefore, a strong line of defense is needed to protect the brain and the central nervous system. Antibodies within the mucus provide this defense mechanism for the immune system. They play an essential role in killing viruses and bacteria that may be present in the breathed air [49].

The axons of the olfactory neurons (also known as primary olfactory neurons) are assembled together into small fascicles, which collectively form the fiber bundles of the *olfactory nerve*. These bundles of olfactory neurons then penetrate through the *cribriform plate* of the *ethmoid bone* to enter the region called the *olfactory bulb*, where they synapse with *secondary olfactory neurons*. There are two kinds of secondary neurons, namely *mitral cells* and *tufted cells*. Mitral cells and tufted cells play an active role in refining and transmitting the olfactory information to the brain through *olfactory tracts*. The complex structures formed as a result of the synapses between primary and secondary neurons are called *glomeruli*. Typically, axons from olfactory neurons with specific types of receptors converge on a specific glomerulus, hinting that there is a coding mechanism between types of receptors and individual glomeruli. Each glomerulus receives approximately 25,000 inputs from receptor neurons, and several from second order neurons.

Also located in the olfactory bulb are *interneurons* called *granule cells*, which are employed in releasing a neurotransmitter to inhibit secondary olfactory neurons for adaptation. The olfactory information carried by the olfactory tracts into the brain terminates at the *olfactory cortex* in the frontal lobe of the brain for final processing.

2.3 Olfactory Physiology

Essentially, olfaction is a solubility process, where the odorant is the solute and the mucus covering the olfactory epithelium is the solvent. Therefore, for olfaction to take place, the odorant must be in a gaseous state, which requires the original substance causing the odor to be a volatile compound. Volatile organic compounds used in this study are therefore all detectable by the olfactory system. The odorant molecules must also be sufficiently water soluble so that they can be dissolved in the nasal cavity [47]. The dissolved odorant then opens various sodium (Na^+), and chloride (Cl^-) channels in the olfactory neuron by binding to odorant binding proteins that are located in the cilia of the neuron. If there are enough odorant molecules, opening these channels causes an influx of positively charges Na^+ and Ca^{++} ions along with an efflux of Cl^- ions, raising the membrane potential from its resting value and creating an adequate depolarization for generating an action potential. The action potential, is then transmitted to the olfactory bulb.

2.3.1 Perireceptor Events

Due to the location of the olfactory receptors, the air entering the nasal cavity during breathing must make a hairpin turn in order to stimulate the receptors. Sniffing, which draws more air into the superior regions of the nasal cavity, is effective for smelling, since it increases the concentration of the odorant molecules received at the sensory receptors.

Processes that affect the entry, exit, and residence time of odorant molecules in the receptor vicinity (the cilia of the olfactory neurons) are collectively known as *perireceptor events.* The most important of these events is the entry of the odorant molecules into the receptor and it is controlled by the odorant binding molecules. The odorant binding molecules serve a number of interrelated purposes. They [49, 52]

- shuttle the odorant molecules towards the chemosensory cilia (odorant receptors),
- initiate the signal transduction process, and
- concentrate the odorant molecule to facilitate their interaction with the cilia.

It is believed that the binding process greatly improves the sensitivity of the olfactory system. This is because, humans are able to sense certain molecules at a concentration of a few parts per trillion, though the individual olfactory receptor neurons can only detect concentrations that are at least 1000 times larger than this amount [49]. As an example, humans can detect 1/25 trillionth of a gram of methylmercaptan, which has a nauseating odor similar to that of rotten cabbage [51]. Therefore, it is usually added at a concentration of 1 part per million to natural gas, an odorless but dangerous gas. Figure 2.2 illustrates the perireceptor events [52].

2.3.2 Odorant Receptors and Olfactory Signal Transduction

Odorant receptors are proteins found in the cilia of the olfactory receptor neurons. They are members of a superfamily of receptor proteins called *G-protein-coupled* receptor superfamily. The specific G-protein that is coupled to odorant receptors is known as *Golf* (*G* protein in *olf*actory receptor).

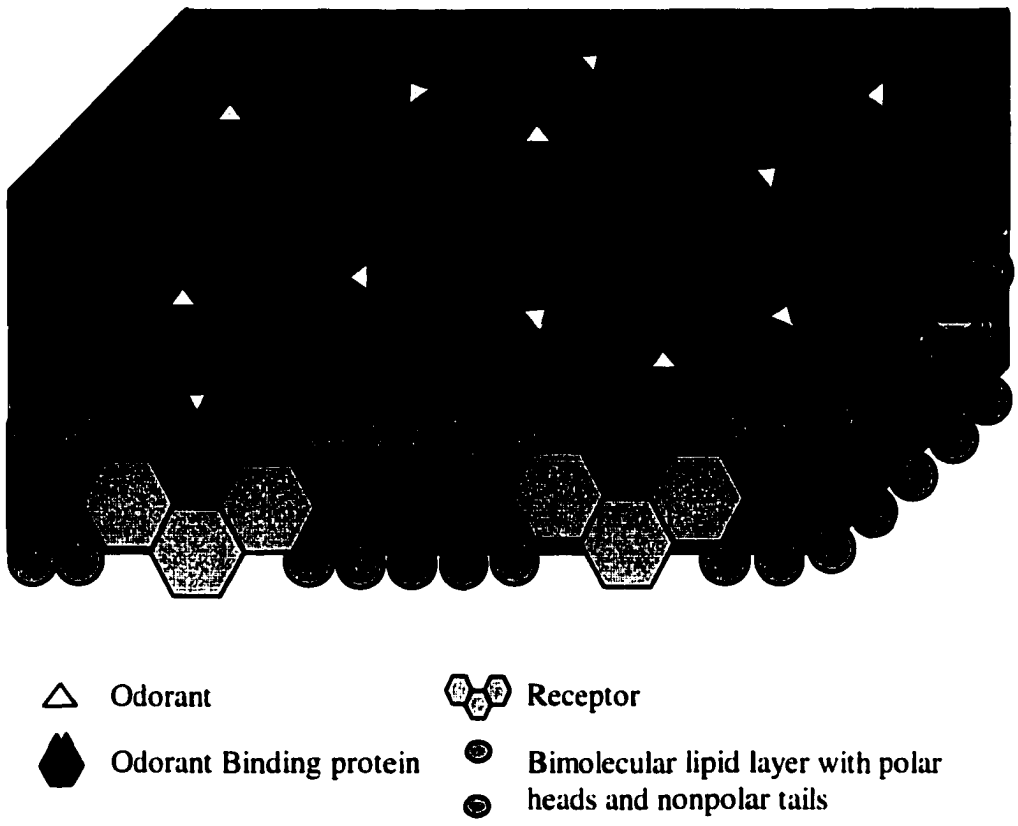


Figure 2.2 Perireceptor events

The odorant binding protein shuttles the odorant molecule to the odorant receptor, and binding of the odorant/odorant binding protein complex to the receptor activates Golf, which in turn leads to the activation of adenylyl cyclase. Adenylyl cyclase is an enzyme that can convert adenosine triphosphate (ATP) to cyclic adenosine monophosphate (cAMP).

ATP is an organic molecule that stores and releases chemical energy for use in body cells, whereas cAMP is an important intracellular second messenger which regulates a variety of cellular effects. In olfactory neurons, cAMP opens Na^+ and Cl^- channels, which results in an influx of positively charged Na^+ and Ca^{++} atoms. The membrane potential starts rising from its resting value of –70mV towards the threshold level, the required voltage for an ac-

tion potential to be fired. In the meantime, the efflux of negatively charged Cl^- atoms further elevates the membrane potential.

In the presence of adequate amount of odorant molecules, the action potential threshold is exceeded and the action potential is fired, carrying the odorant information to the brain. This entire chain of events is known as the *olfactory signal transduction*.

Figure 2.3 illustrates molecular configuration of the olfactory epithelium in the absence of an odorant. Note that all ion channels are closed, and the membrane potential is around -70mV.

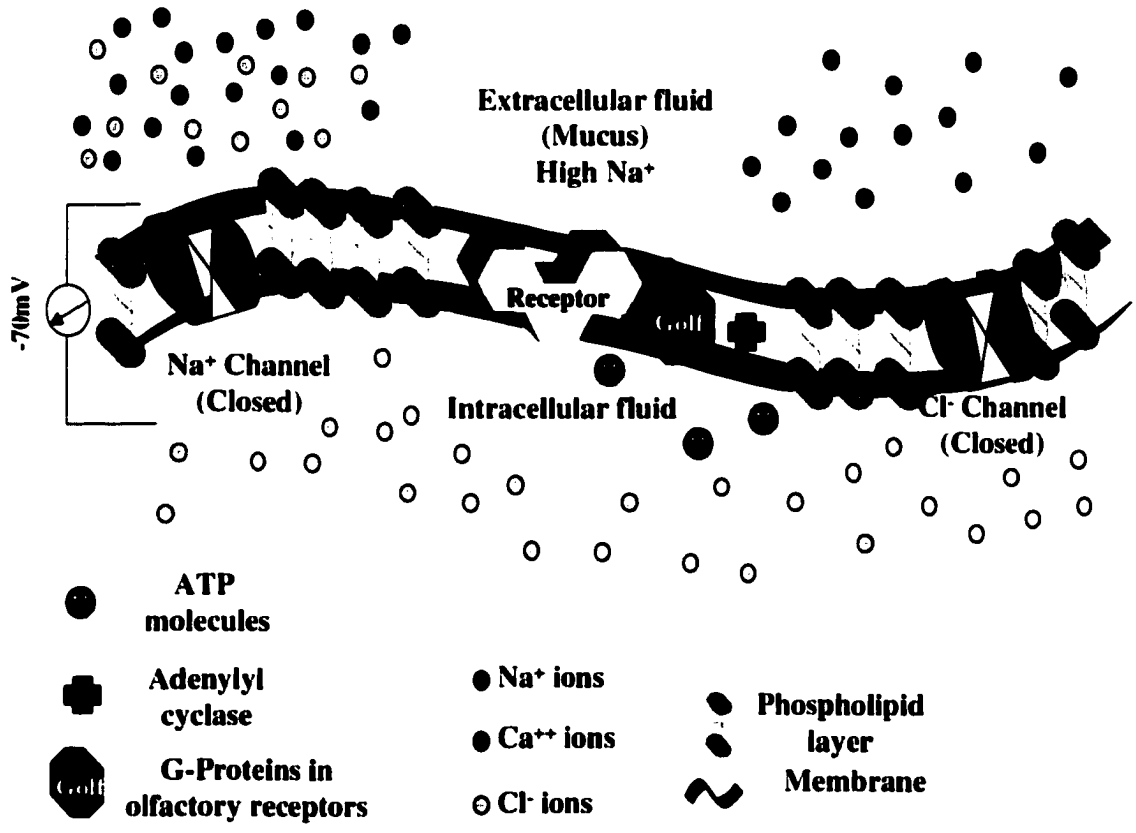


Figure 2.3 Olfactory membrane in the absence of an odorant

Figure 2.4 illustrates the chain of events when an odorant binds to the receptor. Note that Golf activates adenylyl cyclase, which converts ATP to cAMP, causing the ion channels to open. This leads to influx of Na^+ and Ca^{++} ions and an efflux of Cl^- ions, which immediately causes depolarization.

It should be noted that all these events are actually taking place at the cilia of the primary olfactory neurons. Depolarization of the cilia and firing of APs then spread to the entire cell body and to the axon of the neuron.

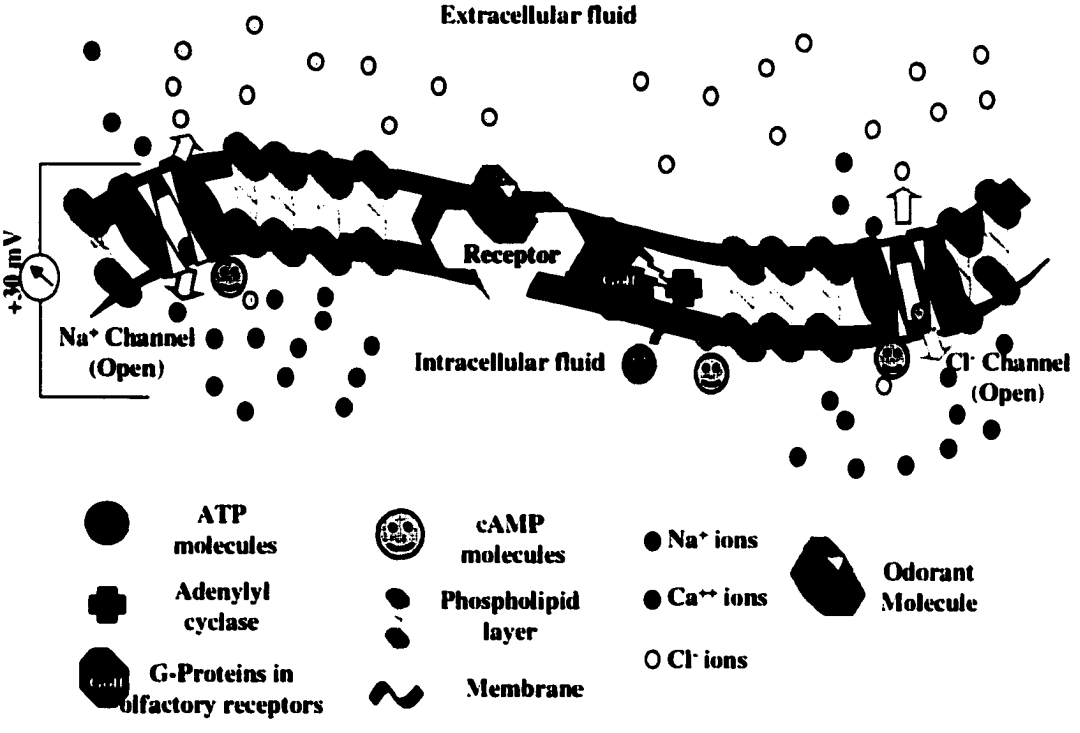


Figure 2.4 Olfactory signal transduction in the presence of an odorant molecule

2.4 Olfactory Pathways

Action potentials generated at the cilia of the primary olfactory neurons propagate towards the axon of the neuron and from there towards the secondary olfactory neurons, which are located in the olfactory bulb. Both types of secondary neurons, mitral and tufted cells, make synapses within the glomeruli to receive the olfactory information from the primary neurons. They also make synapses with granule cells, which are interneurons between afferent (towards the brain) and efferent (from the brain) pathways.

Granule cells complete a long inhibitory feedback loop between the glomeruli and the olfactory cortex of the brain where the olfactory information is processed. The inhibitory nature of this loop is due to a neurotransmitter called gamma aminobutyric acid (GABA) that is released by the granule cells to inhibit mitral and tufted cells, which are employed in relaying the signal from the primary neurons to the olfactory tract. Inhibition of these cells effectively shuts down the afferent signals from the receptors to the brain. Thus, granule cells, receiving information both from the second order neurons and from the brain modify the olfactory information in the olfactory bulb, before it even reaches the brain [51]. This inhibitory feedback loop is associated with the *olfactory adaptation*, which desensitizes the primary neurons to a specific odorant that is continuously being drawn into the nasal cavity.

If not inhibited, mitral and tufted cells send the olfactory information to the olfactory cortex, which is divided into three areas: the *lateral olfactory area*, the *medial olfactory area*, and the *intermediate olfactory area*. Most of the olfactory tract axons terminate in the lateral olfactory area, located at the inferior and medial surface of the frontal lobe of the brain, and therefore the lateral area is considered the primary olfactory area. The lateral olfactory area is responsible for the conscious perception of smell, and the pathway to this region of the brain

has the unique property among all other sensory pathways that it does not make a synaptic relay in the thalamus before reaching the cortical areas. Connections to other cortical regions such as the amygdala and adjacent structures of the limbic system provide visceral and emotional reactions to odors. For example, the nausea due to smelling of a rotten egg, remembering past events and memories upon smelling familiar odors, sexual excitement felt due to specific perfume are all examples of odor-evoked responses related to this pathway.

Olfactory pathways to the medial olfactory area are considered secondary olfactory pathways, and these pathways do make a synaptic relay at the thalamus before reaching the orbitofrontal cortex of the brain. It has been suggested that this secondary (and smaller) pathway may actually have a more direct involvement in the conscious perception of odors then the primary olfactory pathway, and that the primary olfactory pathway is more involved in odor evoked memory and recollection of past events [49].

The intermediate olfactory area is involved in providing the feedback information to the granule cells for the modulation of the olfactory information. [51].

More recently, an additional olfactory pathway has been reported where the olfactory tracts project to the hypothalamus through the accessory olfactory bulb. It appears that the receptors that send information through this route are located in a small pit, called the *vomeronasal organ (VNO)* inside the nasal septum. VNO has been known to exist in lower mammals, particularly in rodents. In humans, however, it was thought that only infants had a VNO, and that it was later lost. Recently, it has been reported that adults do actually have a VNO, albeit not well developed. Much like it does in rodents, VNO provides important olfactory information about nonvolatile odors, such as *pheromones*, which are odorant sub-

stances produced by animals to attract, threaten, or cause other physiologic and/or behavioral changes in other animals [49, 50, 54].

2.5 Sensitivity and Selectivity of Olfactory Receptors

Humans, who are not trained to recognize specific odors can generally identify 10000 different odors, whereas those who are trained, such as wine testers or perfumers can identify an order of magnitude more odors [49]. Therefore, the discrimination of the olfactory system is quite remarkable. However, it is generally believed that these 10000 different odors are a combination of a much smaller number of primary odors, and that the olfactory receptors can actually respond to a very small number of primary odors. The number and identity of these primary odors, however, is an issue of much debate. In fact, researchers have been divided into two main camps: the first group of researchers argues that there are only seven primary odors: floral, musky, camphorous, peppermint, ethereal, pungent (stinging), and putrid (rotten). The second group of researchers argues that humans can actually respond to more than fifty distinct families of odors. More recently, it has been suggested that there are actually over 1000 *smell genes* in the olfactory neurons, each of which encodes a unique receptor protein. Each receptor protein can respond to several distinct odors, and similarly, each odor can bind to several different types of receptors [47]. Considering that there are roughly 80000 to 100000 genes in the human body, about 1% of the human genome is used for olfaction [50].

It is also believed that there is only one type of receptor in each olfactory neuron, and each neuron activates one or two glomeruli, justifying the 1800 glomeruli for 1000 receptors found in the olfactory epithelium. It therefore appears that individual glomeruli are tuned to specific molecular features or odorants. Therefore, recognizing over 10000 odors requires

that odorants stimulate more than one type of receptor at varying degrees and the identification of the odorant is actually done by the brain based on the patterns of glomeruli that have been activated [55].

As mentioned earlier, humans can also detect certain substances at levels of a few parts per trillion, indicating that the sensitivity of the olfactory system is also quite impressive. The sensitivity levels of the receptors vary for different molecules. For example, the human olfactory system can detect 5.83 mg/L of ethyl ether, 3.30 mg/L of chloroform, and 0.0000004 mg/L of methyl mercaptan. However, the same cannot be said for discriminating the difference in the concentrations of odors. For example, for the slightest detection of a concentration change, there must be an at least 30% difference in the concentration levels. This is quite different from the visual discrimination, where a change of 1% in the light intensity can be detected by the human eye [50].

2.6 Towards The Electronic Nose

As summarized in the first chapter, a system that can mimic the mammalian olfactory system by accurately identifying and quantifying odorant molecules has paramount benefits to a wide variety of industries. Traditionally, human experts and dogs have been used in identifying odors; however, training human experts and dogs is very time consuming and costly. Furthermore, they can be subjective, they cannot be used for quantifying the odor, and they certainly cannot be used for identifying hazardous odors. Gas chromatographs and mass spectrometers are also commonly used for identification. These devices are very accurate; however, they are extremely expensive, time consuming, and bulky. Therefore, they cannot be used in real time or in a field setting. Consequently, a faster, cheaper, and portable solu-

tion is of great value. The overwhelming potential benefits of such a system have sparked great interest in medical, chemical and engineering researchers around the world. Various technologies have been developed over the last two decades in an attempt to design a fast, cost-effective, field deployable and accurate electronic nose system. A typical electronic nose system generally includes an array of sensors appropriately chosen for the particular application, a signal processing module, a knowledge base module, a feature extraction module, a pattern classification module and necessary signal/gas transmission media. Figure 2.5 illustrates the schematic of a typical electronic nose system, where the sensors coated with a chemically interactive material (CIM) are used as transducers to convert the chemical information to electrical signals. A sensor processor, such as a frequency counter or a network analyzer, is then used to acquire and measure the electrical signal, followed by a signal processing algorithm to denoise the signals and increase the signal to noise ratio. A feature extraction module then extracts the relevant features from the signal and feeds them to the pattern recognition algorithm, which is pretrained with signals from a knowledge base.

One can think of the following analogy between the human olfactory system and an electronic nose system. The sensors are analogous to the receptor cells. Much like the human olfactory system employs a large number of receptors, an electronic nose system uses an array of sensors. The sensors are coated with different CIMs so that they can respond to different compounds with varying sensitivity and selectivity. Therefore, the coating material can be thought of as the receptor binding proteins. The signal processing and feature extraction modules of a gas sensing system can be compared to the glomeruli of the olfactory system. Finally, the ultimate identification in a gas sensing system is done by a neural network according to output of the signal processing and/or feature extraction module which process the

signature patterns of the various compounds, whereas in the olfactory system the final identification of an odor is carried out by the brain, based on the olfactory information relayed from the glomeruli.

People developing electronic nose systems usually concentrate their efforts on two of the six blocks shown in Figure 2.5: selection of an appropriate sensor, and an appropriate pattern recognition and classification algorithm.

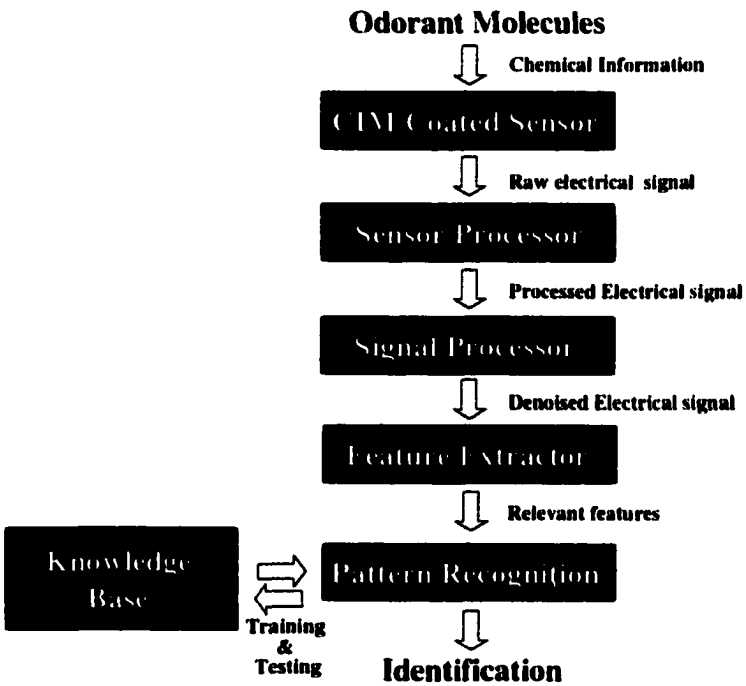


Figure 2.5 Block diagram of a typical electronic nose system

2.6.1 Sensor Technologies for Electronic Noses

Important design issues must be considered in selecting an appropriate sensor technology to use in an electronic nose system. These issues include sensitivity and selectivity, speed of response, cost, size, ability to operate in diverse environments, repeatability, and the ability to clean itself in the absence of odorants, which requires that the sensor should somehow be

flushed of all chemicals after the odorant is identified. There are quite a few sensor technologies available today that satisfy these requirements. Currently available sensor technologies include metal-oxide semiconductors, conductive polymers, conducting oligomers, non-conducting polymers with embedded conductors, surface acoustic wave devices, bulk acoustic wave devices, quartz crystal microbalances, chemical field effect transistors, fiber optic sensors, and discotic liquid crystal sensors [56]. Among these, metal-oxide semiconductors, conducting polymers, surface acoustic wave devices and quartz crystal microbalances have enjoyed more attention than the others.

Metal-oxide (MeO_x) semiconductor gas sensors are based on adsorption of odorant molecules by a metal oxide layer on a semiconductor, followed by subsequent surface interactions which modulate the conductivity of the semiconductor. Silicon dioxide is typically the semiconductor of choice in MeO_x sensors. MeO_x sensors are widely used because they are inexpensive and robust, and they are particularly effective in detecting combustible and hazardous gases. However, they require elevated operating temperatures between 100°C and 600°C. The reproducibility and response time of these devices have recently been improved along with the thin film deposition techniques, whereas sensitivity and selectivity have been enhanced by the addition of catalyst substances (such as palladium) and precise control of operating temperature [57].

Conducting polymer sensors are based on the conductivity changes that take place in the organic semiconductor polymeric materials when they are exposed to volatile chemicals. Conducting polymers have unique adsorptive surfaces that interact with adsorbed volatile chemicals, depending on their shape and size. The change in conductivity depends on the stereochemical parameters of the odorant molecule and the corresponding interaction be-

tween the odorant and the polymer, which causes the polymer to swell. Conducting polymers have fast response, can be easily flushed and they are able to operate at ambient temperatures; however they are very sensitive to humidity [58].

Chemical field effect transistors (ChemFET) sensors consist of a field effect transistor whose gate is coated with a selective coating, typically a polymer. In the presence of an odorant molecule, the coating swells, causing a change in the conductivity of the gate electrode. These sensors have high sensitivity and selectivity, however, having the odorant molecules penetrate the transistor gate constitutes a major difficulty for this sensor [56].

Fiber optic sensors (FOS) employ an optical fiber, which is also coated with a selective coating, typically a fluorescent material. When exposed to odorant molecules, the coating material swells and changes its optical properties, which causes a shift in the frequency of the optical signal transmitted through the fiber [56].

Surface acoustic wave (SAW) sensors and quartz crystal microbalances (QCM) are both piezoelectric devices, whose operation principle is based on a frequency shift in response to an added mass on their surface. These devices are generally coated with polymeric materials chosen according to the chemical properties of the gas to be detected. SAW devices consist of input and output transducers which are deposited on the surface of a piezoelectric material, as shown in Figure 2.6. Between the transducers is a substrate covered by a chemically interactive material (typically a polymer), along which the surface acoustic wave that is generated by the input transducer travel towards the output transducer.

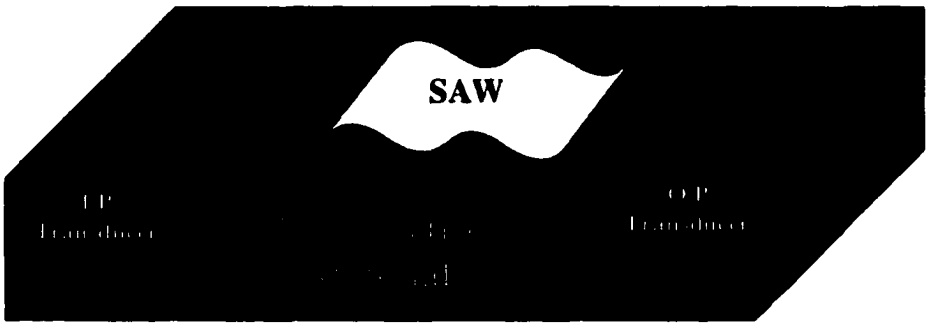


Figure 2.6 Surface acoustic wave transducer

When the odorant material is deposited on the polymer coated surface of the SAW device, the phase velocity as well as the propagation loss of the acoustic wave is altered according to the mass of the deposited amount, and the chemical interaction between the odorant molecule and the polymeric material. In particular, phase velocity can be converted into a frequency shift, which can then be detected as the response of the device to the odorant molecule.

Quartz crystal microbalances (QCMs) work on a similar principle, except the frequency shift that is recorded is the change in the resonant frequency of the piezoelectric crystal when it is exposed to a particular odorant molecule. QCMs, which were used in this study, are described in detail in Chapter 3.

Both piezoelectric type devices are commonly used in gas sensing because they have good sensitivity (down to 0.1 ppm), they can operate at near room temperatures (10°C to 60°C), their selectivity can be controlled by the polymer coating selected according to the odorants that need to be detected, they are inexpensive, easy to prepare and easy to use. These devices are particularly useful in detecting volatile organic compounds.

More recently, hybrid systems that utilize more than one of the above listed technologies started to emerge. One such system is MOSES, the "*MOdular SE*nsor *S*ystem for Gas Sensing and Odor Monitoring"[59]. MOSES combines QCM sensors, SAW devices, and metal oxide sensors, with other modules being under development.

As described above, all sensor technologies involve a selective coating material to be cast on the sensor surface to bind the odorant of interest, and polymers have been the coating of choice for most of them. Despite its preferred conductivity and solubility characteristics (see Chapter 3 for solubility parameters of polymers), polymers are not nearly as selective and/or sensitive as the olfactory receptors in the mammalian olfactory system. Noting that olfactory receptors are simply made of proteins, and that such proteins can be easily extracted from various animals, the natural experiment to try is to use such proteins as coating material. As simple and straightforward as it might seem, this idea was not implemented until Wu used the olfactory receptor proteins (ORPs) of bullfrogs as coating material on QCMs for the detection of various volatile organic compounds [60]. He showed that ORPs produce rapid, reversible, long term and stable responses with sensitivity levels close to that of humans. More information on currently available gas sensors can be found in [61].

2.6.2 Classification Algorithms for Electronic Noses

The second issue of consideration in the design of an electronic nose is the pattern recognition and classification algorithm. No algorithm can approach to the computational power of the brain which can identify over 10000 odors. However, such computational power is typically not necessary for a practical electronic nose application. Usually, a small number of odorants are of particular interest for a typical application, and as described in Chapter 1, a

variety of pattern recognition algorithms have been successfully used in the past. Such algorithms include, but are not limited to, artificial neural networks (ANN), principle component analysis (PCA), cluster analysis (CA), fuzzy logic based algorithms such as fuzzy ARTMAP (FL), discriminant analysis (DA), statistical pattern recognition (SPR), etc. Among these, ANNs and PCA have proven more useful than others.

2.6.3 Commercially Available Electronic Nose Systems

A number of electronic nose systems have been made commercially available within the last decade. Table 2.1 lists some of the more popular commercially available systems (as of March 2000), the sensor technology and the pattern recognition algorithms they employ, and the country of origin for their manufacturers. A more detailed description of these products can be obtained from NOSE (*N*etwork on artificial *O*lfactory *SE*nsing) web page http://nose.uia.ac.be/review/). The following additional acronyms are used in the table: MS: Mass spectrometry, GC: gas chromatography, ND/PR: not disclosed or proprietary, N/A: not applicable, ?: unknown .

Design and development of electronic nose systems is still in its infancy. However, this is an area of active research and progress is constantly being made. Surveys on current research and updates in this area are now available in various electronics and signal processing magazines [62], applied science review magazines [63], as well as in reference handbooks covering a wide area of topics in gas sensing [64].

Table 2.1 Commercially available electronic nose systems[1]

Electronic Nose System	Sensor Technology	# of Sensors	Pattern Recognition Algorithms	Price (US$)	Country of Origin
Airsense Analysis GmbH	MOS	10	ANN, DC, PCA	20,000-43,000	Germany
Fox x000 AlphaMOS	QCM, SAW, CP, MOS	6-24	ANN, DFA, PCA	20,000-100,000	France
AromaScan OsmeTech Inc.	CP	32	ANN, FL	20,000-75,000	U.K.
BH114 Bloodhound Sensors Inc.	CP	14	ANN, CA, DA, PCA	?	U.K.
Cyranose 320 Cyrano Sciences Ltd.	CP	32	PCA	5,000	USA
Enose 5000 Marconi Ltd.	QCM, MOS, CP, SAW	8-28	ANN, DA, PCA	?	U.K.
Znose Electronic Sensor Tech.	SAW, GC	1	SPR	19,500-25,000	USA
QMB6 – HS40XL HKR / Sensorsysteme GmbH	QCM	6	ANN, PCA	?	Germany
MOSES II Lennartz Electronik GmbH	QCM, MOS	16	ANN, PCA	?	Germany
NST 3210 Nordic Sensor Technologies	MOS, FET, QCM	22	ANN, PCA	40,000-60,000	Sweden
OligoSense OligoSense	CP	ND/PR	ND/PR	?	Belgium
SAM Daimler RST Rostock	QCM, SAW, MOS	6-10	ANN, PCA	50,000	Germany
SMart Nose 300 SMart Nose	MS	N/A	DFA, PCA	?	Switzerland
VOCmeter MoTech Sensorik, GmbH	QCM, MOS	8	ANN, PCA	?	Germany
FreshSense Element Ltd.	MOS	ND/PR	ND/PR	?	Iceland
4440B HP – Agilent Technologies	MS	N/A	Various Chemometrics	79,900	USA
VaporLab Sawtek Inc.	SAW	2	ND/PR	5,000	USA

1. The prices are obtained from [62], whereas the rest of the information is compiled from the NOSE web site at http://nose.uia.ac.be/review, and the respective web sites of the manufacturers.

CHAPTER 3

GAS SENSING USING POLYMER COATED PIEZOELECTRIC DEVICES AND THE VOC DATABASE

3.1 Introduction and Overview

In 1880, French scientists Pierre Curie and his brother Jacques Curie observed the strange phenomenon that pressure exerted on surfaces of a quartz material generated an electric potential across the surfaces of the crystal. Equally interesting was the converse of this phenomenon, where an electric field applied to two surfaces of a quartz crystal caused deformations on these surfaces. They named this phenomenon as the *piezoelectric effect*, where the word was derived from the Greek word πιεζο (piezo) which means *to press* or *exert pressure*. The ability of piezoelectric materials to convert mechanical deformations and vibrations into electrical potentials, and conversely, convert voltage into mechanical motions, allowed them to be used in a wide variety of applications such as mechanical to electrical (and vice versa) transducers. In particular, they have been widely used in oscillator circuits, high frequency amplifiers and microphones. The first reported use of piezoelectric materials as acoustic transducers in ultrasonic applications was in 1917 [61].

In late 1950s and early 1960s, King showed that quartz piezoelectric crystal coated with an appropriate coating material could be used as a microbalance to detect gases or liquids accumulated on the surface of the coating. Hence, quartz crystal microbalances (QCMs) were born as sorption detectors [65]. Around the same time, Sauerbrey provided the theoretical foundation of the piezoelectric sorption detector by formulating the expression of the shift in

resonance frequency of the piezoelectric crystal in response to a mechanical mass deposited on its surface [66]. The Sauerbrey equation for QCMs, which describes the frequency change, Δf, caused by a deposited mass Δm is given as

$$\Delta f = \frac{1}{\rho N} \cdot f \cdot \frac{\Delta m}{A} \tag{3.1}$$

where ρ is the quartz density, N is a crystal related constant, f is the fundamental resonant frequency of the uncoated crystal, A is the active surface area. This equation is often approximated as [67].

$$\Delta f = -2.3 \times 10^6 \cdot f^2 \cdot \frac{\Delta m}{A} \tag{3.2}$$

for QCMs.

From 1970s to 1990s, the technology for manufacturing and using piezoelectric crystals for gas sensing applications has progressed at an exponential rate. Today a variety of highly sophisticated piezoelectric sensor devices are commercially available, and piezoelectric acoustic wave sensors now comprise a versatile class of chemical sensors for various gas sensing applications.

The particular application that is discussed in this dissertation is the detection of volatile organic compounds (VOCs), which are organic compounds that can readily evaporate at room temperature and pressure. VOCs used in this study were originally in liquid phase (analyte) from which vapors of VOCs were obtained. Therefore, the terms VOC, analyte and vapor are used interchangeably in the following discussion.

For sensing applications, a coating film is cast on the surface of the QCM. This layer can bind a VOC of interest, altering the resonant frequency of the device, ideally, in proportion

to the added mass. The chemical sensor typically constitutes of an array of several crystals, each coated with a different coating. Through monitoring the response pattern of an array of coatings, this design is aimed at improving identification, which is hampered by the limited selectivity and varying sensitivity of individual sensors. The response pattern then serves as a signature for a given VOC.

3.2 The Quartz Crystal Microbalance

Quartz crystal microbalances (QCMs) constitute a subgroup of acoustic wave devices known as *thickness-shear mode (TSM)* devices. Acoustic devices consist of a piezoelectric material with one or more metal transducers on their surfaces. Acoustic waves at ultrasonic frequencies are launched into the material from these transducers serving as electrodes. The acoustic waves launched into the material have particles which move normally (perpendicular) to the direction of wave propagation, and hence they are called *transverse waves* or *shear waves*.

For gas sensing applications, the surface of the crystal is first coated with gold, which serves as the metal transducer, to obtain electrical contacts (electrodes). The device is then coated with a material that is sensitive to the analyte to be detected as illustrated in Figure 3.1. The interaction of the coating material with the analyte perturbs various parameters of the acoustic wave such as the wave velocity and the resonant frequency. The amount of perturbation on the shear wave depends on the thickness of the coating (among other things as described later in this chapter), from which the name "thickness shear mode" devices was derived.

In one type of QCM, the bare crystal is about 1 cm in diameter and resonates at approximately 9 MHz, whereas another type of QCM is approximately 25 mm in diameter and resonates at 25 MHz. The coating material is typically a polymer film to bind the molecules of the VOC of interest. The VOC in vapor form (the solute) is then solved in the coating material (the solvent/sorbent). The interaction between the coating and the VOC results in

1. mass accumulation on the piezoelectric crystal
2. swelling of the coating and changes in the shear modulus (a measure of stiffness) of the coating [68, 69]

both of which alter the resonant frequency of the device. The exact interaction between the analyte and the coating depends on the viscoelastic properties of the coating material, and the solubility parameters.

Viscoelastic properties of the coating material include thermal expansion and film resonance effects. These properties are measured by the *shear modulus*, which is a measure of material stiffness. For example, an increase in the temperature causes the polymer to swell, which in turn causes the material to soften and hence reduce its shear modulus. This effect is known as thermal expansion and can cause significant shift (decrease) in the resonant frequency of the acoustic device. Continually increasing the temperature, however, causes a sudden increase in the device resonant frequency, and this effect is known as the film resonance effect. Film resonance effect is due to lower surface of the film adhering to the crystal more strongly than the upper surface of the film, particularly at high temperatures. At the lower surface, the polymer moves synchronously with the crystal, whereas at the upper surface the polymer movement lags behind. When the phase lag due to this nonsyncronous motion on the upper and lower surfaces of the coating reaches ninety degrees, film resonance is

said to occur, responsible for the sudden increase in the device resonance frequency [69]. Film resonance effects are particularly prominent in thicker coatings.

The solubility parameters play an even more substantial role in the coating – analyte interactions and therefore they are discussed in detail in the next section.

In designing gas sensors, the aim is to choose coating materials such that the sensitivity and selectivity of the sensors are maximized for the analytes of interest. Both viscoelastic properties and solubility parameters of a coating must be considered for intelligent selection of coatings to ensure improved sensitivity and selectivity. The issues that need to be considered in coating selection are discussed in the next section, where an introductory review of solubility interactions between the analyte and the coating material is also included. This review is compiled primarily from [44, 45, 46, 68, 69, 70, 71, 72, 73, 74, 75]. Detailed information on QCM sensors, including design issues, circuit equivalents, sensitivity analyses in various media, can be found in [68, 69, 72, 76, 77, 78].

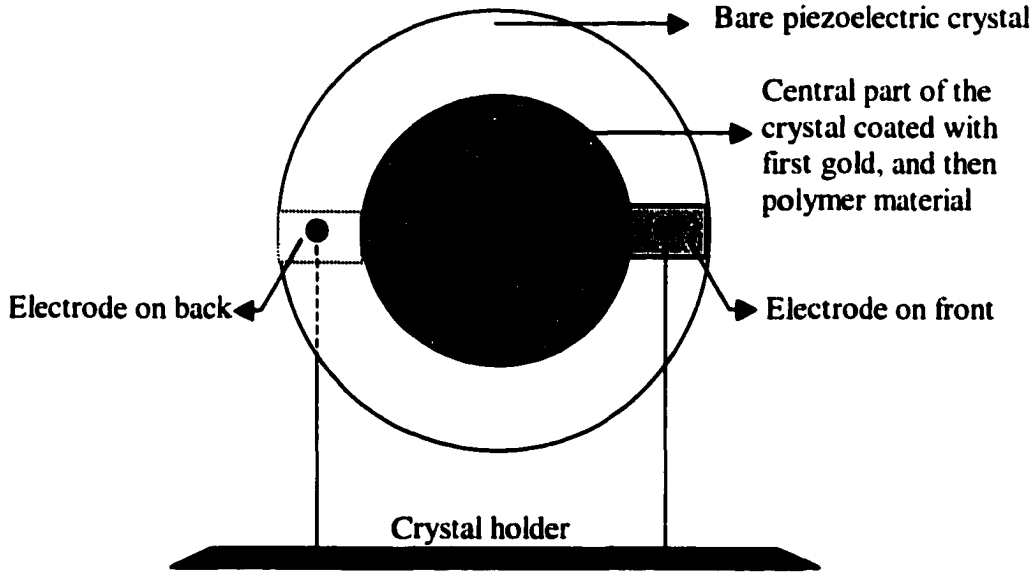


Figure 3.1 Schematic diagram of a QCM

3.3 Coating Selection Considerations

3.3.1 Sensitivity and Selectivity

Two major issues in chemical detectors are sensitivity and selectivity. Sensitivity, S, refers to the device being able to detect analytes at low (trace) concentrations, and it is defined as the incremental signal change occurring in response to an incremental change in analyte concentration. The signal measured is typically the change in resonant frequency Δf, and therefore, the sensitivity can be defined as

$$S = \frac{\Delta f}{\Delta p} \tag{3.3}$$

where Δp is the change in the concentration of the analyte. Sensitivity is given in Hz/ppm (parts per million). Sensitivity is directly proportional to the coating thickness, as thicker coatings respond with larger frequency shifts.

Selectivity refers to the ability of the device to selectively detect the analyte of interest in the presence of other materials, and/or the ability to give a significantly different response to different vapors, so that they can all be identified.

The sensitivity and selectivity of each sensor to a particular analyte can be controlled by strategically selecting the coating material with specific chemical and physical properties such that certain solubility interactions between the coating and the analyte are maximized. Sensitivity and selectivity of a coating increases with the strength of the solubility interactions that take place between the coating and the vapor. A number of physical parameters as well as chemical properties of the coatings play a central role in determining the sensitivity and selectivity of a coating.

3.3.2 Physical Parameters Affecting Sensor Response

Physical parameters that must be taken into consideration can be summarized as follows. It should be noted that these properties are interrelated with the viscoelastic properties of the coating material.

1. ***Thickness of the coating:*** Increasing the coating thickness increases the amount of vapor that can be collected at the surface, and hence increases sensitivity. However, thicker coatings also increase the resistance of the coating, and causes film resonance due to phase lag (described in Section 3.2), which in turn results in attenuation of the surface energy on the device and consequently very slow response times.
2. ***Softness / Stiffness of the coating:*** In general soft coatings have better response times, and usually result in reversible processes. However, they are also lossy, and cause attenuation in the oscillator circuits used, preventing the measurement of the resonant frequency. Therefore, soft coatings cannot be made very thick. On the other hand, stiff coatings are not lossy; and therefore, they can be made thicker. However, they have slow response and irreversibility problems.
3. ***Reversibility:*** Reversibility assures that the device can be repeatedly flushed of the analyte and exposed back to the same or a different analyte or concentration. Typically, weak interactions between the analyte and the coating material allow good reversibility, but in return, they hamper the sensitivity and selectivity. Strong interactions improve sensitivity and selectivity, but may cause irreversibility, or very slow reversibility.
4. ***Operation temperature:*** One of the major parameters of polymeric coating materials is their *glass transition temperature*, a temperature at which they transform from an

amorphous and *elastomeric* state to a *glassy* and *crystalline* state. Above their glass transition temperature *(Tg),* these materials become softer and provide faster responses. Therefore, if operating temperature is, for example, room temperature, then coatings with *Tg* values less than 25°C should be chosen. On the other hand, it should be noted that as temperature increases, sorption decreases, and hence sensitivity and selectivity decrease.

3.3.3 Intermolecular Interactions Affecting Solubility

Sensitivity and selectivity depends on the strength of the sorption, which depends on the strength of the solubility interaction, which in turn depends on the solubility properties of the solute and the solvent. *Likes like likes* is the general rule of thumb in solubility, where the objective is to maximize specific interactions between the analyte (solute) and the coating (solvent/sorbent). The specific solubility interactions relevant to chemical sensing are the following intermolecular vapor-coating interactions [44, 46]:

1. ***Induced dipole / induced dipole (dispersion interaction):*** Also known as London forces, these are the interactions between primarily nonpolar molecules, and they contribute significantly to the sorption of all vapors by organic polymers. They occur due to movements of unevenly distributed electrons causing instantaneous / momentary dipoles.
2. ***Dipole / induced dipole (dipole induction):*** These are the interactions of uncharged, nondipolar, but polarizable (when exposed to an electric field) molecules with dipoles. The strength of dipole induction depends on the polarizability of the nonpolar molecule, but they are generally weak.

3. ***Dipole / dipole (dipole orientation):*** These are electrostatic interactions which involve the attraction between the positively and negatively charged regions of dipolar molecules. These interactions can be very dominant when strongly dipolar molecules interact, and they can be even stronger for certain orientations of the dipoles.
4. ***Hydrogen bonding:*** These interactions are special cases of very strong dipole – dipole interactions and occur typically between a hydrogen atom and a small but highly electronegative element, such as fluorine, oxygen or nitrogen. They are important in many chemical and biochemical processes, and they involve the directional interaction between a hydrogen-bond acidic site and a hydrogen-bond basic site.

The first two of these interactions are non-oriented and the last two are oriented interactions. The first three are also known as Van der Waals interactions, although some chemists use this term specifically for dispersion interactions. A quantitative measure of vapor molecules to participate in these interactions can be obtained through various models using *solubility parameters*. These parameters measure the solubility properties of the vapor.

3.3.4 Linear Solvation Energy Relationships and Solvation Parameters

To quantify the total absorption, *the partition coefficient, K,* is defined as the ratio of the solute concentration in the sorbent, C_s, to the solute concentration in vapor, C_v, that is

$$K = C_s / C_v \tag{3.4}$$

The partition coefficient is a thermodynamic parameter which measures the equilibrium distribution of the solute molecules between the vapor phase and solvent phase. This is illustrated in Figure 3.2.

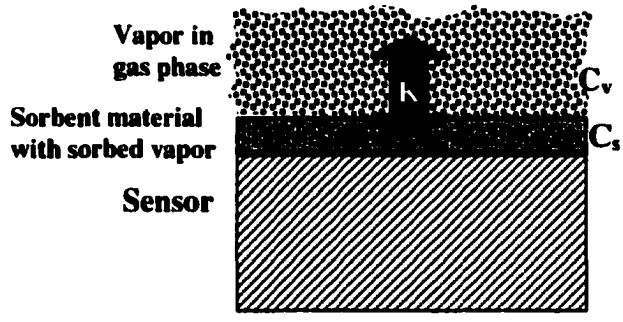


Figure 3.2 Absorption of a vapor from the gas phase into the sorbent phase (From [46])

The partition coefficient is useful when applied to piezoelectric detectors because it provides a direct relationship to the frequency shift caused by the mass loading effects of the sorption of the vapor. For example, for surface acoustic wave sensors,

$$\Delta f_v = \Delta f_s C_v K / \rho \tag{3.5}$$

where Δf_s is the frequency shift caused by the application of the coating material to the bare crystal, Δfv is the frequency shift caused by vapor sorption, ρ is the density of the coating and K is the partition coefficient. It should be noted that Equation 3.5 is an empirical expression. In general, the amount of solute that can be absorbed is large if the K value is large. Therefore, higher partition coefficients lead to better sensitivity of the coating.

Various models have been proposed that relate the partition coefficient to various solubility properties of the solutes. Simple models such as the boiling point model that relates K to vapor's boiling point, or the Hildebrand solubility parameter method have proven to be of limited use, but they have been far from optimized [45]. A more sophisticated model, called the linear solvation free energy relationship (LSER), has been shown to be the most accurate model of the relationship between K and solubility properties of the solute [45]. In this model, the partition coefficient is defined empirically in terms of the above listed solubility interactions. A number of parameters that characterize the solubility properties of solutes

have been devised in an attempt to measure the abilities of solute molecules to participate in the above listed interactions. These parameters are collectively referred to as *solvation parameters*:

α^H: Vapor hydrogen bond donation term, measures the hydrogen bond acidity.

β^H: Vapor hydrogen bond acceptance term, measures the hydrogen bond basicity.

R: Polarizability term measuring the ability of a solute to interact with the solvent through n and π electron pairs.

π^H: Vapor dipolarity-polarizability term, measuring the ability of a molecule to stabilize a neighboring charge or dipole.

L^{16}: Ostwald solubility coefficient, provides a measure of cavity formation and dispersion interactions (it is the partition coefficient of the solute in hexadecane at 25°C).

The LSER used for sorption processes has the form

$$\log K = c + rR_2 + s\pi_2^H + a\alpha_2^H + b\beta_2^H + l\log L^{16} \tag{3.6}$$

where the indices $_2$ denote that these parameters refer to the solute, the parameters r, s, a, b and l characterize the complementary properties of the coating material, and c is a regression constant. The complementary parameters, along with solvation parameters, measure the strengths of various interactions in affecting the solubility in a given solvent. In practical terms, a large s value corresponds to strong sorbent dipolarity, whereas a large a value corresponds to increased ability of the sorbent to accept hydrogen bonds from hydrogen bond donating groups of the vapor. Similarly, a large b value corresponds to increased ability of sor-

bent to donate hydrogen bonds to hydrogen bond accepting groups in the vapor. The parameter r measures the ability of the sorbent to interact with the solute's n and π electrons.

When combined with solvation parameters, $s\pi_2^H$ is the polarity term, rR_2 is the polarizability term, $a\alpha_2^H$ is the hydrogen bonding term where the solute is a hydrogen-bond acid, $b\beta_2^H$ is the hydrogen bonding term, where the solute is a hydrogen-bond base, and $l \log L^{16}$ is the combined dispersion interaction and cavity term.

It is worth noting that not all terms in the LSER equation are important for all cases. In some cases, certain terms can be eliminated if the corresponding interactions do not have substantial contributions to the overall solubility. For example, hydrogen-bond acidic sorbents are relatively uncommon, and therefore the $b\beta_2^H$ can be omitted for all solvents that are not hydrogen bond acidic.

As seen from the LSER equation, the chemical properties of both the solute and of the solvent play significant roles in determining the solubility properties and, therefore, in determining selectivity and sensitivity. Therefore, solvation parameters of analytes of interest, along with the corresponding parameters of the coating must be considered before the coating selections are made. The following generalizations can be made regarding various groups of analytes.

Alkanes are only capable of dispersion interactions, since they have zero values of α^H, β^H, R, and π^H, and nonzero L^{16} values. All aliphatic alcohols (in which carbon atoms are linked through open chains) are moderate hydrogen bond acids and hydrogen bond bases, phenols are more acidic, whereas fluoroalcohols and fluorophenols are very strong hydrogen-bond acids, since fluoro substitution reduces the basicity of alcohols. On the other hand,

ethers, ketones, esters, nitriles are all moderate bases, whereas amines, amides, sulfoxides, N-oxides are strong bases. As mentioned above, most bases are dipolar, and in general, stronger bases have stronger dipolarity, except for amines. Aliphatic amines, have strong basicity, but little dipolarity. Conversely, nitriles are strongly dipolar though only moderately basic.

Aromatic molecules, such as benzene, toluene and xylene all have lone pairs of π electrons, resulting in strong polarizability. Chlorinated and brominated molecules, as well as aromatic molecules are strongly polarizable, due to significant R and π^H values. Fluorinated molecules, however, are not very polarizable [46].

LSER parameters for a variety of polymers can be found in [45, 46, 70].

3.3.5 VOCs of Interest

Twelve VOCs (analytes) of interest were chosen to represent a wide variety of functional groups. They constitute a wide diversity and span all major interactions. These VOCs and their properties whose chemical formulas are given in Appendix I, were as follows [44, 46, 71, 75]:

1. Acetonitrile (ACN): High dipolarity and basicity.
2. Acetone (AC): Ketone, moderate base, dipolar.
3. Methyethyketone (MEK): Ketone, moderate base, dipolar.
4. Octane (OC) : Alkane, nondipolar, only dispersion interaction.
5. Hexane (HX): Alkane, nonpolar, only dispersion reaction.
6. Ethanol (ET): Aliphatic alcohol, dipolar, hydrogen bonding, moderate acid/base.
7. Methanol (ME): Aliphatic alcohol, dipolar, hydrogen bonding, moderate acid/base.
8. Xylene (XL): Aromatic hydrocarbon, polarizable.

9. Toluene (TL): Aromatic hydrocarbon, polarizable.

10. 1,1,1-Trichloroethane (TCA): Chlorinated alkane, polarizable.

11. Trichloroethylene (TCE): Chlorinated, polarizable.

12. 1,2-Dichloroethane (DCA): Chlorinated, polarizable.

Solvation parameters of the above listed VOCs, as well as those for several thousand analytes, can be found in [79].

3.3.6 Designing the Coating Material

Recall that the objective is two folds: The first goal is to obtain the largest sensitivity possible to the target analyte. In order to achieve this goal, the coating material must be designed to have properties complementary to those of the analyte, such that all possible interactions between the coating and the vapor are maximized. Maximizing all possible interactions also maximizes the partition coefficient according to Equation 3.6, which in turn maximizes the sensor response according to Equation 3.5. For example, a good coating for the sorption of a dipolar basic vapor would be a polymer that is also dipolar, but hydrogen bond acidic. The second goal is to obtain the best selectivity for a target analyte, which has conflicting requirements to those of obtaining best sensitivity. In particular, the attempt should be to design a coating that will maximize a single interaction that is favored by the target analyte and minimize all others. If an array of coatings is used, then there should be at least one coating for each solubility interaction, in addition to coatings that are specifically designed to maximize the sensitivity to the target analytes.

Unfortunately, it is practically impossible to find materials that will only have one type of interaction. In fact, all organic materials, for example, go through dispersion interactions, and

all basic materials are also dipolar and vice versa. The practical approach is to pick a solubility interaction, and incorporate this interaction through a suitable coating such that all other interactions are minimized. The procedure is repeated for all interactions.

Other issues affecting coating selection are how fast the analyte is to be detected (response time), at what concentration levels it is to be detected, and whether it is to be detected reversibly, all of which affect the chemical and physical properties of the coating to be selected. Also of particular importance is whether other compounds exist in the environment, which could potentially generate strong signals from the sensors. If a potential interference with a specific compound is strongly suspected, then it is usually necessary to include a coating that is especially selective for the offending compound. If the interfering reaction is with the water vapor, *hydrophobic* coatings can be chosen that do not like water.

Polymers are one group of materials that are especially suitable as sensor coatings due to their favorable physical and chemical properties. First of all polymers are non-volatile, which allow them to stay on the sensor once they are applied. They can be easily applied by a variety of methods such as spin coating, air brushing, adhesion, etc. Furthermore, the glass transition temperatures of most polymers are typically low enough to ensure that the sensors operating at room temperatures are operating well above their glass transition temperatures. This property makes the polymers soft enough to provide rapid and reversible responses (at the expense of stronger attenuation of the acoustic wave energy). Another important property is that, as long as bond-making or bond-breaking interactions do not occur, polymer coated sensors are reversible.

3.3.7 Designing a Sensor Array

As mentioned earlier, finding a single coating that will undergo only one kind of interaction is not possible (for example, all polymers are capable of dispersion interactions), nor is it possible to find a coating that will strongly participate with all types of interactions. Furthermore, using a single sensor, it is not possible to tell whether a response is due to low concentration of an analyte to which the coating is very sensitive, or due to high concentration of an analyte to which the coating is not very sensitive. The problem multiplies when there is more than one analyte to be detected, or when there are interfering vapors. All these scenarios are convincing evidence that an array of sensors should be used rather than a single sensor, for identification and quantification of analytes. The question is then how to select an array of coatings that will detect the desired analytes and not respond to undesired ones.

If there are multiple analytes to be detected, the strategy is then to include coatings such that each coating will strongly interact with a single group of vapors, but not with others, effectively forcing each sensor to highlight one type of solubility interaction, or a combination of solubility interactions. In other words, the sensors should be chosen to maximize the diversity in the array response to various analytes.

In this study a set of six sensors was used. They were coated with the coatings shown in Table 3.1. The polymers were applied using a variety of techniques, such as painting, air brushing, submerging, etc. based on the properties of polymer. These coatings were selected to maximize various solubility interactions. For example, APZ and PIB can easily undergo dispersion interactions with hexane and octane, but they do not interact much with ketones or alcohols. DEGA does not interact with alkanes, but it does interact with alcohols, probably

Table 3.1 Coatings used for this study

Coating Name	Abbreviation	Thickness,(μm)	App. Δf (kHz)
Apiezon (grease, not a polymer)	APZ	1.1	24
Poly(isobutylene)	PIB	2.7	60
Poly(diethyleneglycoladipate)	DEGA	0.6	14
Sol-gel	SG	1.1	24
Poly(siloxane)	OV275	0.6	14
Poly(diphenoxylphosphorazene)	PDPP	2.2	49

via dipole-dipole or hydrogen bonding interactions [70, 75]. Polarity and hydrogen bonding (basicity) are also important for OV275 and PDPP.

Another important property of these coatings that were considered in the selection criteria was that they are all *hydropobic* (does not like water) material, eliminating the interfering interaction with water vapor.

However, even the most carefully selected group of coatings may not be adequate to provide necessary discriminatory information, particularly when the analytes of interest include multiple gases from the same chemical family. For example, the response patterns of many polymers are very similar to ethanol and methanol, or to toluene and xylene.

Figure 3.3 illustrates this problem. In Figure 3.3, normalized frequency shifts of QCMs with the six coatings to four different VOCs are shown. The vertical axis is the frequency shift, and each bar in the horizontal axis represent the response of one polymer coating. Note that the response patterns to toluene and xylene are very similar, since both chemicals are aromatic hydrocarbons. Similarly, responses to both ethanol and methanol are also similar, since both of them are aliphatic alcohols.

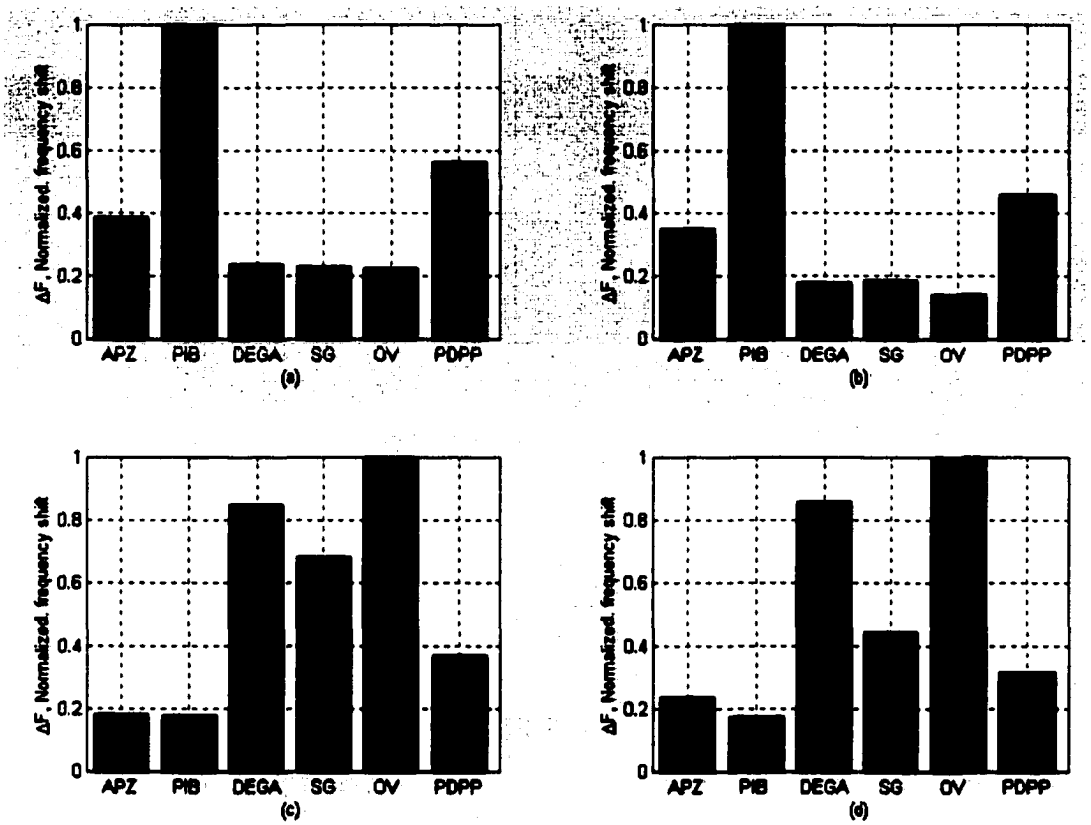


Figure 3.3 Responses of six coatings to (a) toluene, (b) xylene, (c) ethanol, (d) methanol

The limited information obtained from the sensors, the signature patterns of the analytes, can be most efficiently utilized by the use of a suitable pattern recognition technique. Pattern recognition techniques, such as neural networks, Bayesian classifiers, Fisher's linear discriminant, principal component analysis, etc. can be very effective in recognizing a large number of different response types, even when the differences in responses to different analytes may be unnoticeable to human perception. However, it should always be remembered that the performance of any pattern recognition technique is limited by the quality and the quantity of the information provided by the sensors.

To a pattern recognition algorithm, a response pattern provided by n sensors is simply a point in the *n-dimensional space* called the *feature space*. For example, in Figure 3.3, each set of six responses by the sensors for any given VOC is the signature pattern of that VOC, and constitutes a point in six dimensional space. The individual response of each sensor then constitutes one feature of the response pattern. If responses of sensors to different analytes are similar to each other, then these responses will be interpreted as points which are very closely packed in the feature space, and hence it will be difficult to identify which point corresponds to which analyte. On the other extreme, if we could construct an array in which each coating would respond to one and only one analyte of interest, then the responses would fall on orthogonal axes on the n-dimensional feature space, and the identification of the analytes from their responses would be trivial.

Intuitively, an array of sensors, coated with a set of polymers that accentuate a variety of combinations of solubility interactions will certainly provide better information to a pattern recognition scheme than a set of sensors in which all sensors undergo the same interaction. Therefore, a diverse set of coatings with strong, selective and uncorrelated responses to different analytes will facilitate the classification (identification) task of the pattern recognition scheme. In such a case, response patterns for different analytes will be represented by points distant to each other in the feature space, and hence it will be easy to identify which points represent which analytes.

Despite best efforts, finding a good set of coatings that would easily identify all analytes of interests is not easy, or may not even be possible. This is particularly true when a number of mixtures of analytes are present in the environment, and the goal is to identify each mixture with its individual components. In such cases, advanced pattern recognition techniques

specific to the problem need to be developed. Chapter 4 of this dissertation introduces three such pattern recognition techniques for the identification of mixtures of VOCs.

Another important problem is the selection of a subset of the *most* useful coatings for the classification problem at hand when there are a number of possibly useful coatings available. When analytical methods summarized above fail to make such decisions, heuristic optimization techniques can be employed to select the best subset of coatings. This problem is addressed in Chapter 5 of this dissertation where two techniques are introduced for the sensor array optimization problem.

The experimental setup and data acquisition system are described in the remaining sections of this chapter. Also presented in this chapter are the selectivity challenges that the responses provided to the pattern recognition scheme.

3.4 Experimental Setup

The sensors used in this study were ~9MHz QCMs purchased from Standard Crystals. Cr/Au contacts were evaporated onto the quartz by means of a resistive heating evaporator (Edwards Coating System E306A). Dilute solutions of polymers, typically 20μL of 0.3-3 % w/w, were used to spin coat the gold surfaces at 2000-5000 rpm. The sensors were then dried at ~65 °C for 15-24 hours. The thicknesses of coatings were calculated from the frequency shifts caused by the coating [80]. The coated QCMs were then mounted in a sealed test fixture, which housed six sensors.

The vapor generation system consisted of a carrier gas, typically dry nitrogen flowing at a constant flow rate of 200 sccm, a gas stream module, and a pair of three-way switchable valves, leading into the test fixture housing the six crystals. The gas stream module included

a VOC module and a reference module that served to establish the baseline. The switchable three-way valves were used to maintain continuous flow of the reference gas and the VOC, so that an uninterrupted steady state was maintained. The final output was a constant flow rate with periodical exposure to known levels of VOC. This flow was obtained by means of calibrated mass flow controllers (Tylan® general FC-280 AV) and conventional gas bubblers containing the analytes. The bubblers were composed of two connected compartments. The gas carrier bubbled through the solution in the first compartment, supplying the vapor, whereas the second analyte-containing compartment served as a headspace equilibrator. This resulted in a vapor stream of variable flow rate for different concentrations, but of constant level at each concentration. The vapor at various concentrations was further diluted with nitrogen to generate the final output mixtures of desired concentrations.

The sensors were exposed automatically to the vapor stream by means of computer controlled three-way valves and a MKS® multi-gas controller model 147B that controlled the mass flow controllers. Polyethylene and Teflon® tubings together with stainless steel or brass valves were used, with only Teflon and stainless steel being exposed to the analytes. Experiments were performed at ambient temperature. Repeated measurements indicated reproducibility of the collected data with insignificant variations, within experimental error, due to small temperature fluctuations. The frequency response was monitored using a Hewlett Packard® HP8753C network analyzer, interfaced to an IEEE 488 card installed in a PC, and HP8516A resonator-measurement software. Real time data were displayed and saved. The data were then analyzed to obtain frequency shifts (relative to the baseline) vs. VOC concentration lines. Typical noise levels (standard deviations of the baseline) for the QCMs were 0.1 Hz. Figure 3.4 depicts the overall schematic of the experimental setup.

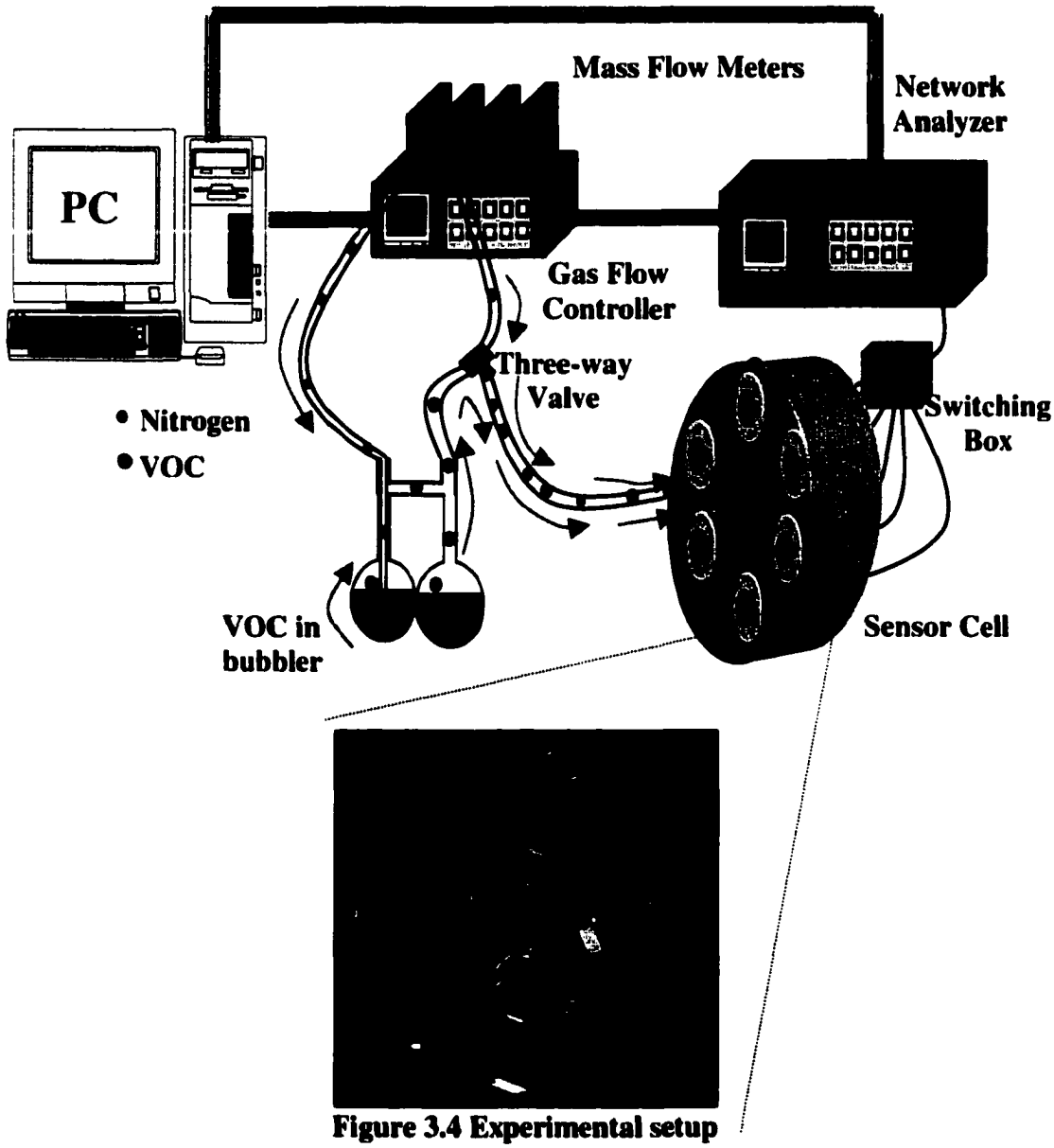


Figure 3.4 Experimental setup

Figure 3.5 illustrates a typical response pattern of the previously listed six coatings to toluene at seven different concentrations. Sensors were exposed to toluene at the shown concentrations in a random order, for a duration of 30 minutes each. After each toluene exposure, sensors were exposed to dry nitrogen for an additional 30 minutes to flush toluene

molecules from the sensor surfaces. Figure 3.5 clearly illustrates the reversibility of the coatings, as the sensor responses returned back to baseline levels during dry nitrogen exposure. Also note from Figure 3.5 that the amount of frequency change is proportional to the concentration of the analyte. In fact, this proportionality is linear for all VOCs used in this study for the concentration range of interest (50~1000ppm), and the proportionality constant constitutes the sensitivity of the sensor for that analyte, as defined in Equation 3.3.

As another example, Figure 3.6 illustrates this linear relationship for TCE, where the horizontal axis is the concentration of the TCE and the vertical axis is the response of the sensors as frequency shifts from the baseline value. Note that the slope of each line defines the sensitivity of the coating for that analyte (TCE, in this case).

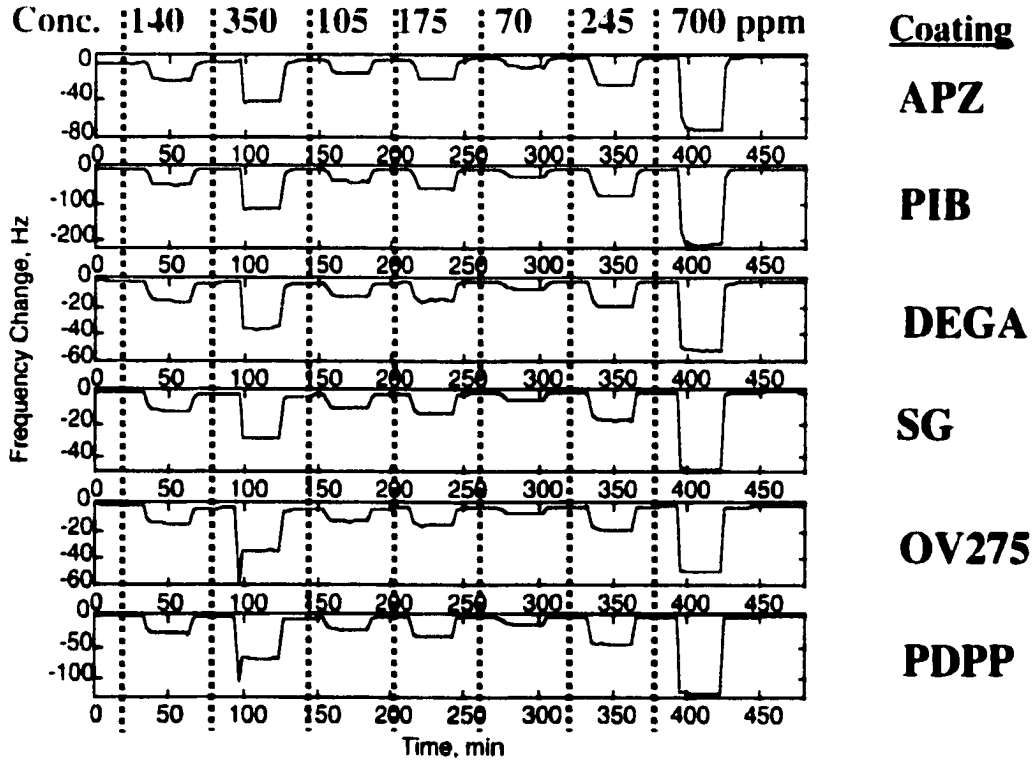


Figure 3.5 Typical response patterns of the six coatings to toluene

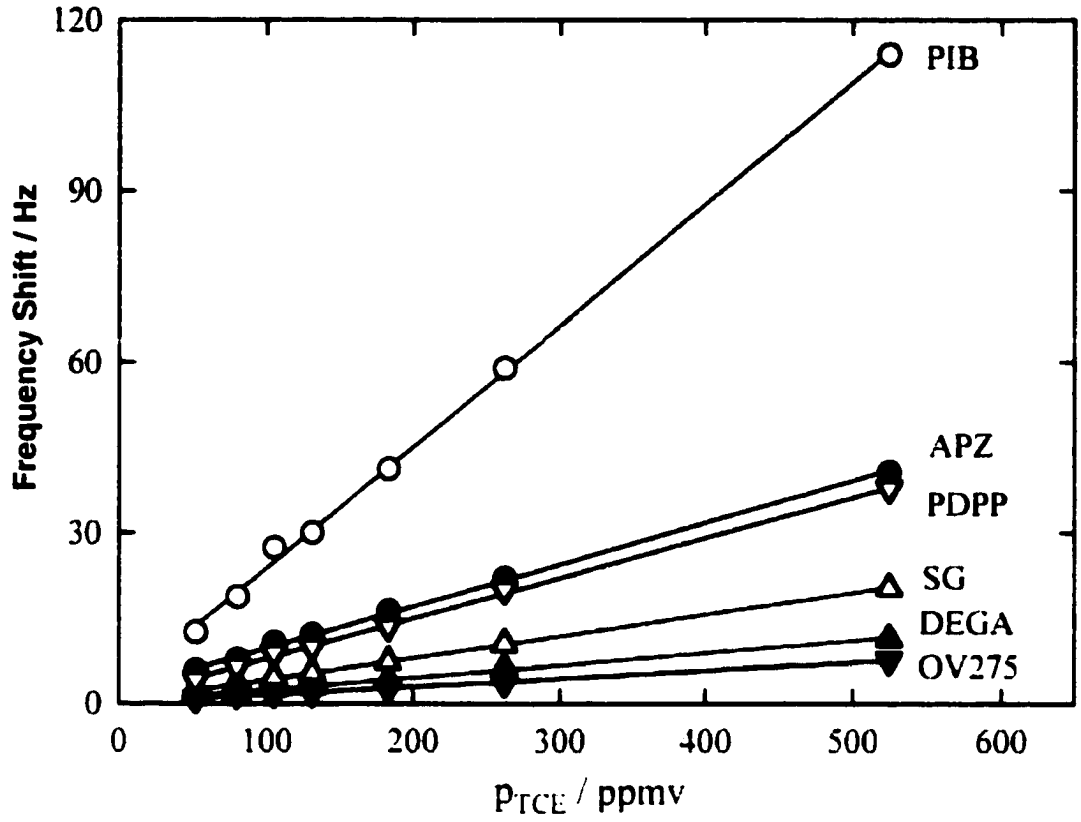


Figure 3.6 Linearity of the responses with concentration

3.5 Identification of Individual VOCs

Individual identification of these 12 VOCs was the initial goal of this project. To our surprise, this turned out to be a very simple task for a single hidden layer multilayer perceptron (MLP) neural network with 6 input, 20 hidden and 12 output nodes. Out of 84 patterns, seven for each VOC for 12 VOCs, 30 were used to train the network and the rest were used to evaluate the performance. All 54 patterns were classified correctly.

3.6 Problems in Identification of VOCs in Binary Mixtures

Identification of binary mixtures of VOCs, was not as simple as the identification of individual VOCs. Mixture of VOCs constitutes a significant roadblock to the identification of the individual components in the mixture, particularly when a dominant VOC is present in the mixture. The responses of sensors to other VOCs then becomes partially, or in some cases completely, masked by the response to the dominant VOC. Furthermore, certain combinations of VOCs produce almost identical responses at different concentrations. For example, the responses of sensors to a mixture of VOC_A and VOC_B at concentrations $[VOC_A]$ and $[VOC_B]$, might be very similar to the response of sensors to a mixture of VOC_C and VOC_D at concentrations $[VOC_C]$ and $[VOC_D]$, respectively. Therefore, all responses were normalized with respect to concentration, so that the concentration information was removed from all responses. Furthermore, the response patterns of xylene, toluene, and TCE at different concentrations were particularly similar to each other, and responses to these VOCs were significantly larger than the responses to other VOCs tested. These phenomena give rise to two related problems. The similarities of responses to mixtures with different dominant VOCs render identification of the *dominant VOCs* difficult, but the very presence of a dominant VOC makes identification of a *secondary VOC* difficult.

An example illustrating the first problem is the case XL & MEK and TL & HX mixtures, where XL and TL are the dominant VOCs. The original responses to these mixtures with 700 ppm of each VOC, as well as two sets of normalized responses are given in Table 3.2.

In the first normalization scheme shown in Table 3.2(Norm.1), all sensor responses (sensor outputs) were divided by the maximum frequency response in the array. In the second normalization scheme (Norm.2), each response was divided by the square root of the sum of

the square of all responses. Both of these normalization schemes were tried since both are commonly used in signal processing, and depending on the particular application, one of them might be more advantageous than the other. As seen from Table 3.2 (and Figure 3.7), the second normalization scheme is more beneficial in this case, since it allows changes in the maximum response. Note that with the first normalization scheme, the maximum response is always mapped to one. We therefore used the second normalization scheme. Throughout the rest of this chapter, normalization will always refer to the second scheme.

Figure 3.7 illustrates the effects of these normalization schemes on response patterns. The plots on the left are responses of the sensor array to the mixture of XL&MEK. The plots on the right are responses to the mixture of TL&HX. As we can see from Table 3.2 and Figure 3.7, the responses of these sensors to two different mixtures result in very similar patterns.

Figure 3.8 illustrates the second problem, where the presence of dominant VOCs masks the responses to secondary VOCs. Responses to four different mixtures of xylene with a

Table 3.2 Comparing original and normalized responses of two mixtures

	XL&MEK Original (Hz)	TL&HX Original (Hz)	XL&MEK Norm-1	TL&HX Norm-1	XL&MEK Norm-2	TL&HX Norm-2
APZ	290	95	0.36	0.36	0.30	0.29
PIB	793	264	1.00	1.00	0.81	0.81
DEGA	154	54	0.19	0.20	0.16	0.17
SG	172	60	0.22	0.23	0.18	0.18
OV	138	51	0.17	0.19	0.14	0.16
PDPP	411	139	0.52	0.53	0.42	0.42

second VOC are compared. Coatings are listed along the horizontal axes and the normalized sensor responses (in frequency changes) are plotted along the vertical axes.

It should be noted that in all four mixtures shown in Figure 3.8, the concentration of xylene was 150 parts per million (ppm), and the concentration of the secondary VOC was 700 ppm. Despite the significantly larger concentration of the secondary VOCs relative to that of xylene, their effects were almost completely masked by the responses to xylene.

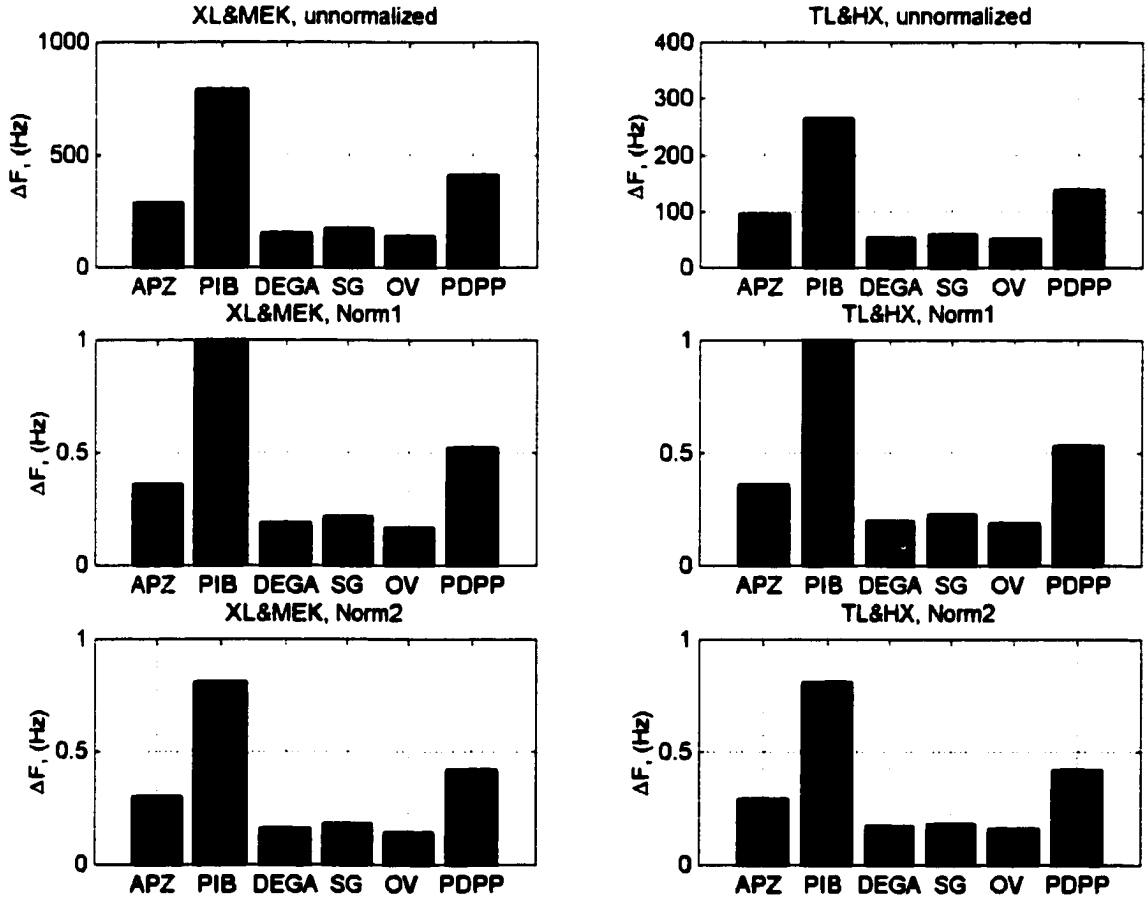


Figure 3.7 Comparing normalized and non-normalized responses of two mixtures

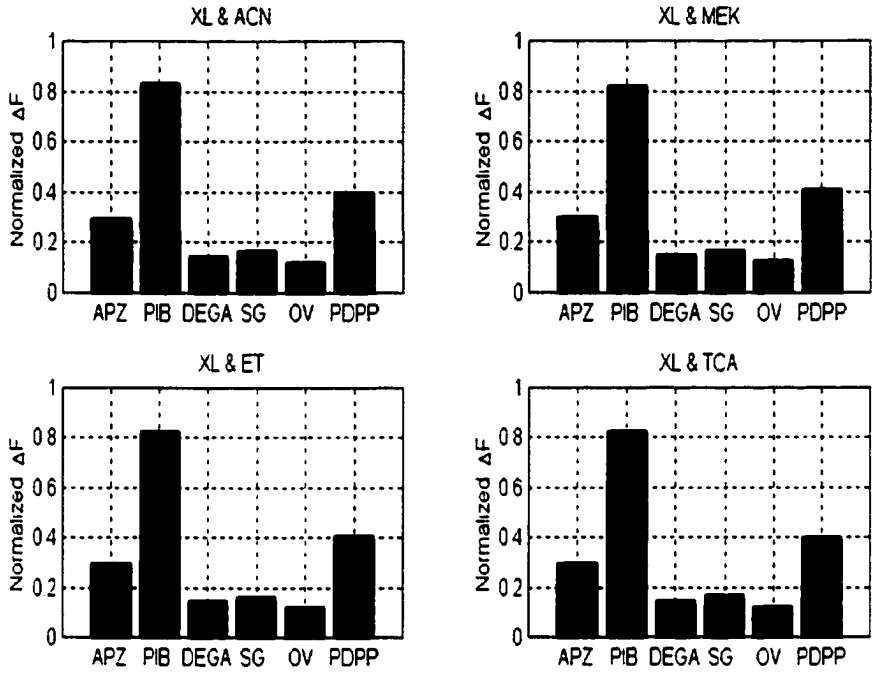


Figure 3.8 Responses of six coatings to various mixtures of xylene

Sixteen different concentration combinations of the following mixtures were considered:

Octane	Xylene	Toluene	TCE	Ethanol
OC & ACN	XL & ACN	TL & ACN	TCE & TCA	ET & ACN
OC & ET	XL & ET	TL & ET	TCE & MEK	ET & MEK
OC & MEK	XL & MEK	TL & MEK	TCE & TL	ET & HX
OC & TL	XL & HX	TL & HX	TCE & ET	ET & TCA
OC & TCA	XL & TCA	TL & TCA	TCE & HX	

Each column represents mixtures of one of the five dominant VOCs, (OC, XL, TL, TCE and ET) with one of other secondary VOCs. Sensors were exposed to these mixtures at all combinations of 150, 300, 500 and 700 parts per million (ppm), giving 16 combinations of concentrations for each of the 24 mixtures listed above (that is, 150 and 150, 150 and 300,

150 and 500, 150 and 700, 300 and 150,..., 700 and 500, 700 and 700 ppm). Twenty-four mixtures of 16 different combinations of concentrations generated the 384-pattern database used in this study.

It should come with no surprise that no neural network architecture or learning paradigm was able to converge for identifying even the dominant VOCs in the mixture, let alone the individual VOCs forming the mixture, and Figure 3.7 and 3.8 clearly demonstrates why this is the case.

These figures also illustrate the need for an effective methodology to classify patterns that look very much alike. Such a classification algorithm must be able to extract the subtle differences between the patterns. In the pattern recognition terminology, such patterns are referred to as *overlapping class distributions* in the pattern space. This is because when plotted in the *d*- dimensional space, these patterns form overlapping clusters.

Researchers working in the gas sensing areas have been collaborating with those working in signal processing and pattern recognition to solve this problem, and various approaches have been proposed. Most approaches have been limited to using standard pattern recognition methods with fine-tuning certain parameters to the specific problem at hand.

As mentioned in Chapter 1, most standard approaches have been one of, or a combination of, principal component analysis, various neural network architectures, discriminant analysis, statistical pattern recognition schemes, etc. However, all of these techniques assume that the data come from a well behaving distribution, and simply attempt to classify or cluster the given data. A number of popular feature extraction and preconditioning schemes are being used to condition the data, however, blind use of these schemes, such as normalizing, Fou-

rier/wavelet transform, denoising, etc. by themselves do not necessarily improve the separability of the data.

The difficulty of the classification problem can be significantly reduced, if the patterns are carefully preprocessed to improve the separability of the data, by augmenting the subtle difference among the patterns. Three methods, specifically designed for enhancing pattern separability are proposed, described, and compared in the next chapter.

CHAPTER 4

ENHANCING PATTERN SEPARABILITY

4.1 Introduction

Pattern recognition, the problem of identifying a multidimensional pattern into one of the prescribed classes, is of great importance in a variety of applications. All areas of engineering, economic and financial analysis, oceanography and seismology, forensic sciences, food sciences, medicine and other biological sciences, are just a few of areas where pattern recognition has found applications. Researchers in all these fields have been working on developing faster, more accurate, more noise tolerant pattern recognition algorithms for decades, as a result of which various schemes have been devised. These schemes include, discriminant based algorithms [81], statistical pattern recognition algorithms (such as Bayes classifiers) [82], Fourier descriptors, syntactic algorithms [83], fuzzy logic algorithms [84, 85, 86, 87], neural networks [85, 86, 87, 88, 89, 90], and more recently support vector machines [91, 92] among many others. Detailed information on these and various other pattern recognition techniques can be obtained from [93, 94].

The ideal case in a pattern recognition problem is to be able draw linear lines among patterns of different classes in the multidimensional pattern space. The simplest problem can be formulated as follows: Consider a set of n-dimensional patterns $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, where each pattern $\mathbf{x}_i$ belongs to one of two classes, labeled as "0" or "1". These patterns are considered *linearly separable* if there exists a linear line defined by $\mathbf{w}^T \cdot \mathbf{x}_i = 0$ separating class "1" patterns from class "0" patterns, through

$$\mathbf{w}_T \mathbf{x}_i \begin{cases} < 0 \, if \, \mathbf{x}_i \in class \, 0 \\ > 0 \, if \, \mathbf{x}_i \in class \, 1 \end{cases} \tag{4.1}$$

The idea can be easily extended to multiclass cases, where **w** then describes a hyper plane. The problem is that most practical applications generate patterns that cannot be linearly separated, or even patterns whose classes form overlapping clusters. The field of pattern recognition is therefore devoted to developing algorithms that can classify patterns that are not linearly separable. Various approaches have been proposed, including generating nonlinear decision surfaces, as neural networks do, or somehow transforming the database through nonlinear mappings so that the data can be linearly separated, as support vector machines do. When transforming the data into linearly separable classes is difficult, or not possible, various heuristic approaches that target increasing intercluster distances have been tried. For example, Chandrasekaran *et al.* have used feature projection as a preprocessing algorithm to increase the separability of the data, where features are mapped to a higher dimension. They have also tried to relieve the burden on the classifier, by converting an n-class problem into n two class problems [95]. Diamantaras *et al.* have used a different approach where they continuously monitored the patterns, which are sequentially fed into a single McCulloch and Pitts neuron (perceptron), to detect each pattern that violates linear separability. They were able to flag any such pattern by carefully observing the weight change, and skip any linear separability violating patterns to be dealt with later. In the second round, all flagged patterns are clustered among each other into linearly separable clusters, further flagging still troublesome patterns. By combining all such single neurons, they claim that their algorithm can learn all convex, but nonlinearly separable classes [96, 97].

Another interesting approach has been developed by Osbourn *et al.* where they use the concept of *region of influence* for clustering data. The method, called *visually-empirical region of influence pattern recognition (VERI-PR)*, is based on computing k-dimensional neighbors of each (k-dimensional) pattern [98]. However, the neighbors are not determined using any of the standard distance metrics, but rather via an empirically determined shape. This shape, called the VERI shape roughly resembles a dumbbell, as two semicircles are combined through V- shaped lines, and at the center of each circle lies a square. The VERI shape is illustrated in Figure 4.1 along with its rotated and scaled versions.

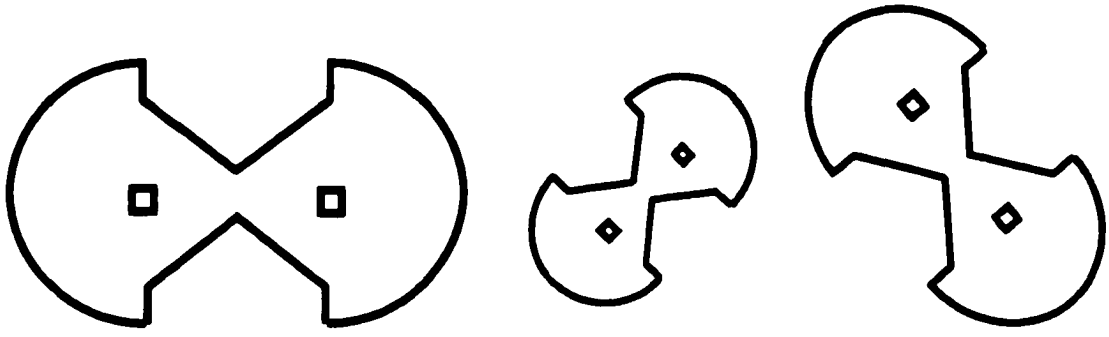


Figure 4.1 Rotated and scaled versions of the VERI shape

The algorithm compares each data point with all other data points one by one, by placing the two data points to be compared on top of the squares of the template VERI shape. The VERI shape is then scaled and rotated to see if any of the data points remaining in the dataset fall within the VERI shape. If no third data point falls within the VERI shape, the two data points compared are placed in the same cluster. Otherwise, those two points do not belong the same cluster. The algorithm has no input parameters other than the data to be clustered, as the VERI shape is fixed and built-in to the algorithm. The VERI-PR algorithm was originally developed for VOC identification, but the authors claim that it is applicable to a variety of pattern classification problems. A detailed description of the algorithm is available from the

Sandia National Laboratories website [99], whereas the application of the algorithm to VOC data, as well as how it can be used for optimum feature selection can be found in [100].

One of the recent developments in pattern recognition involves support vector machines (SVMs) which create a linear separating plane to create a pattern classifier. The main strength of SVMs is that they are able to do this using a minimum amount of data. The algorithm keeps track of those patterns which are closest to the decision boundary between the classes, and tries to maximize the margin separating the decision boundary hyperplane and the selected patterns. These patterns, which determine the decision boundary, are called the support vectors. SVM is an algorithm for separating linearly separable patterns; however, it can easily be extended to handle nonlinearly separable classes. When patterns are nonlinearly separable in the original feature space, SVMs use a kernel to transform the data from its original space into a higher dimensional space where the data is linearly separable. The performances of SVMs are very much dependent on the kernel chosen, and unfortunately, there are no known methods to consistently choose the most appropriate kernel for the problem at hand. However, various kernels have been tried and reported to work well on various types of problems. The kernels that are most often used include the Gaussian radial basis function kernel [91,101], hyperbolic tangent kernel [91,102] and the polynomial kernel [91].

Many of these methods, however, do not specifically target increasing the intercluster distances between patterns belonging to different classes, and those, which indirectly increase intercluster distances do so by increasing the dimensionality of the problem as well. In this chapter three alternative approaches are proposed, all of which specifically target increasing the intercluster distances to enhance pattern separability. In the first approach, a fuzzy inference system is built where the separation of patterns is achieved through strategic

selection of membership functions. In the second approach, intercluster distances are increased through *feature range stretching (FRS)*, a scheme inspired by companding algorithms for communications and the histogram equalization (contrast stretching) for image processing. One disadvantage of FRS processing is that it also increases intracluster distances. Finally, in the third approach, intercluster distances are increased, without increasing intracluster distances, through a *nonlinear cluster transformation (NCT)* which is learned by a generalized regression neural network (GRNN).

4.2 Fuzzy Inference Systems for Enhancing Pattern Separability

4.2.1 Background

Supervised neural networks and fuzzy inference systems are two common methods used in pattern recognition when representative training data are available. Supervised neural networks are capable of learning a mapping function between the samples of feature vectors (sensor responses) and their corresponding classes (VOCs), through a highly nonlinear and massively parallel structure that resembles the structure of the nervous system [88, 89, 90]. A number of neural network architectures and training algorithms have been developed over the years, each of which provides a near optimum solution to various classification problems. Among these, the multilayer perceptron with backpropagation learning rule has enjoyed considerable success in a large range of classification problems. However, for the database under consideration, no MLP was able to identify even the dominant VOC, let alone secondary components of the VOC mixtures. The identification of the dominant VOC using a fuzzy inference system, followed by determining the secondary VOC by a subsequent MLP was therefore chosen as the design strategy.

Fuzzy logic approaches have also enjoyed considerable success among pattern recognition researchers [84, 85, 86, 87]. At the heart of these approaches is a fuzzy inference system (FIS) that classifies patterns using a predetermined set of IF/THEN rules. Unlike a neural network which uses crisp numerical values for computation, FIS uses *fuzzified linguistic values*. This is achieved using a selection of fuzzy sets, such as "small", "large", "very large", and defining membership functions (MFs), $\mu_{SMALL}(x)$, $\mu_{LARGE}(x)$, etc. on these sets. Typical functions used for this purpose include triangular, trapezoidal, Gaussian, and bell shaped membership functions. The system first decides how much each input x belongs to a membership function and assigns a membership value $\mu_A(x)$ A = {Small, Large, Very large, etc}. Based on these membership values, the system then decides how much each rule is satisfied. The mechanics of an FIS is detailed through an example in the following paragraphs. The FIS designed in this study, FNOSE (fuzzy nose), was fine tuned for dominant VOC identification. The secondary VOC identification was achieved by using a neural network, hence a neurofuzzy approach. The overall block diagram of a typical FIS is illustrated in Figure 4.2.

4.2.2 Membership Function Selection and Fuzzification

The first step involved in an FIS is *fuzzification*, the conversion of numerical values to linguistic values. Usually, most fuzzy systems work on normalized data, and therefore the numerical values that need to be converted into linguistic values fall into the [0 1] range. Each sensor response constitutes an input to the FNOSE, and therefore each sensor (APZ, PIB, DEGA, SG, OV275 or PDPP) is considered as an input variable. The fuzzification step of FNOSE converted the normalized numerical values of sensor responses into linguistic variables through previously defined six MFs. Corresponding linguistic variables (fuzzy sets)

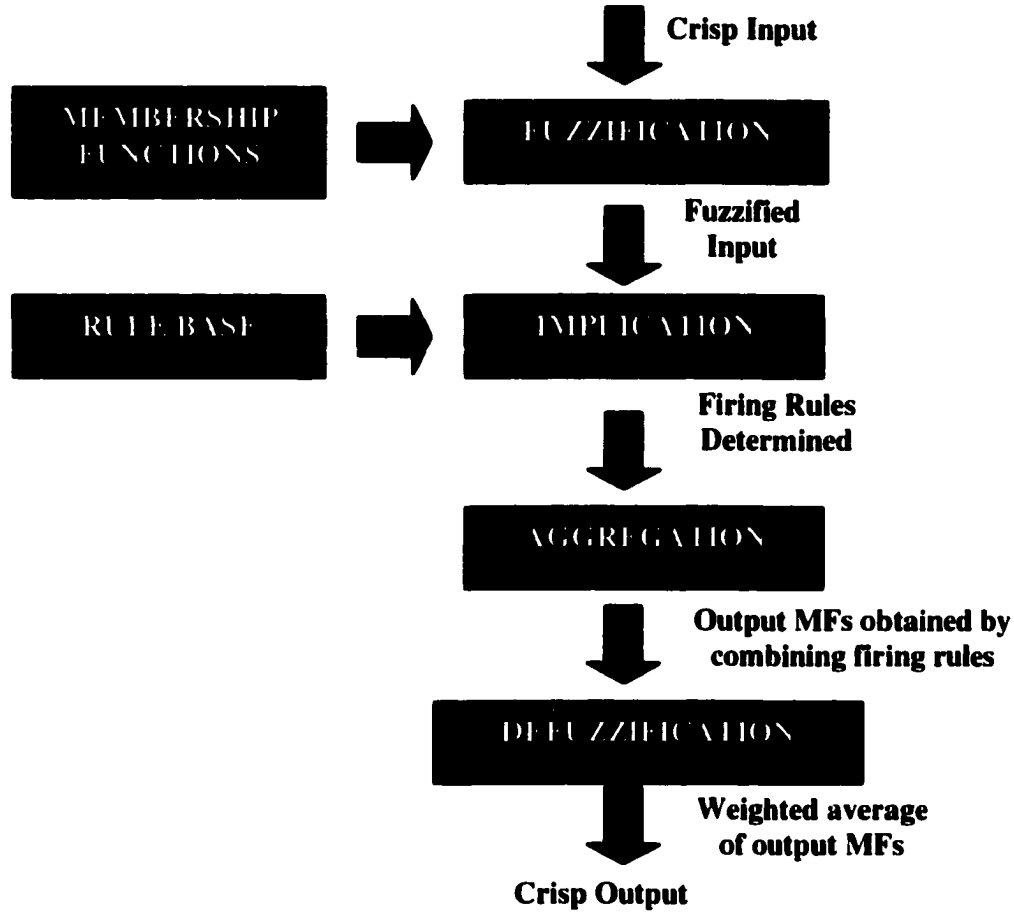


Figure 4.2 Block diagram of a typical fuzzy inference system

were defined as VS (very small), S (small), M (medium), L (large), VL (very large), and XL (extra large).

In our initial design of FNOSE, trapezoidal MFs were placed at equal distances from each other for all sensors, as typically done in generic FIS designs. For example, all sensor outputs in the [0 0.2] range were considered very small, all sensor outputs in the [0.2 0.4] range were considered as small, and so forth. It was soon realized that this approach was far from being optimal, since the individual ranges of sensor outputs were completely ignored.

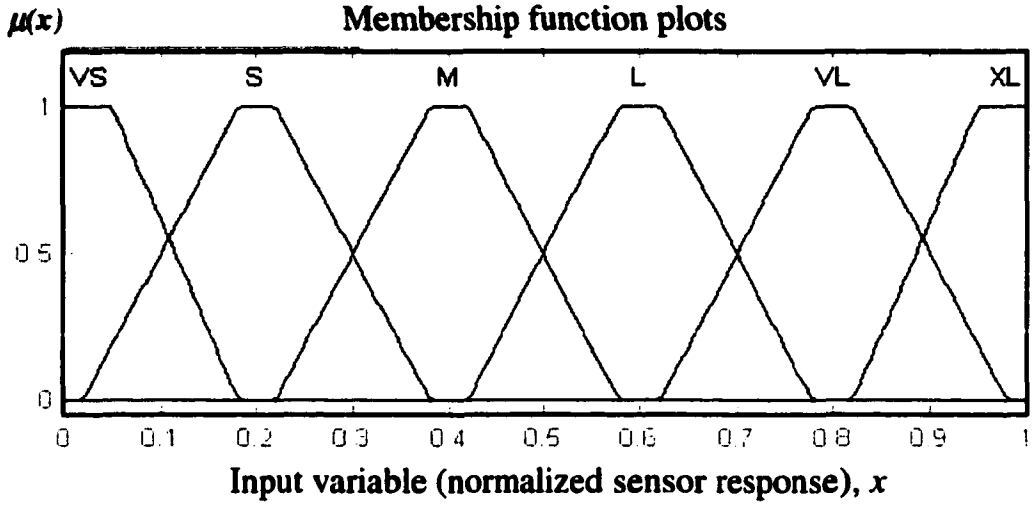


Figure 4.3 Initial selection of membership functions

Figure 4.3 illustrates this inefficient selection of membership functions, where all membership functions were placed at (approximately) equal distances from each other, and the same selection of membership functions was used for all sensors.

A better selection of MFs must consider the dynamic ranges of each input variable (sensor). Consequently, six trapezoidal MFs were placed along the [0 1] range according to the *dynamic range* of individual sensor responses for each sensor. As a first example, Figure 4.4 illustrates the histogram of APZ responses of the 384-pattern database. The horizontal axis shows the APZ responses and the vertical axis shows the frequency of appearance of these responses. As seen from this histogram, the [0.25 0.33] interval constitutes the effective dynamic range for this sensor. Since almost no APZ response falls outside of this range, MFs placed outside this range provide no information to the FIS. Therefore, this range was divided into six intervals (for six MFs) such that maximum discriminatory information can be obtained from the placement of the MFs for this sensor.

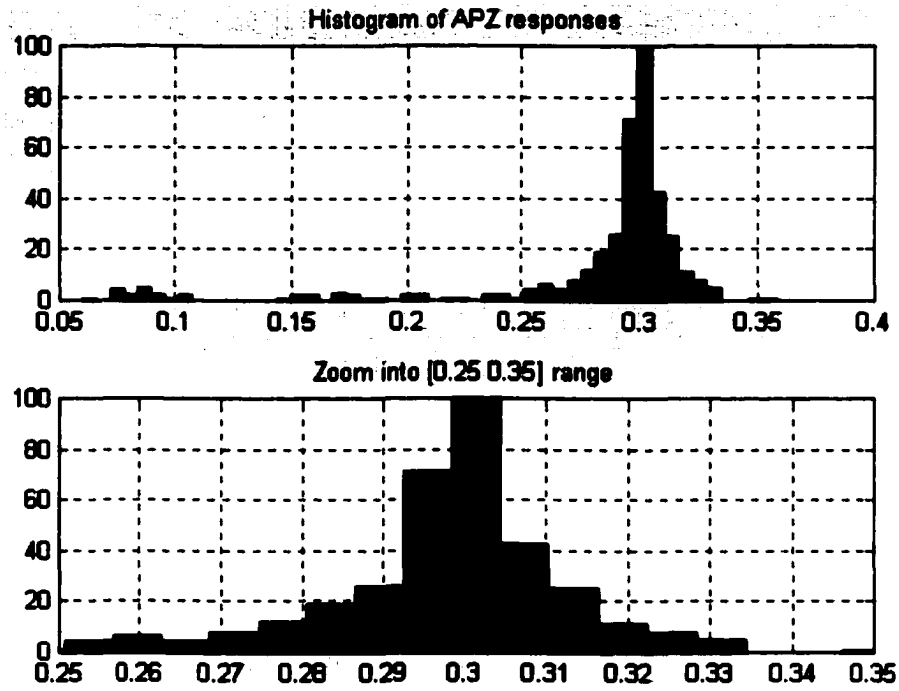


Figure 4.4 Histogram of APZ responses

Figure 4.5 illustrates how APZ membership functions were placed in the effective dynamic range of the APZ sensor. As seen in Figure 4.5, all normalized APZ responses less than 0.26 were assigned to the *VS* membership function, those in [0.27 0.28] and [0.28 0.29] intervals were assigned to *S* and *M* membership functions, respectively, and so forth. Note that there is some overlap among the membership functions, indicating that certain values belong to two membership functions with varying degrees. For example, the value 0.27 equally belongs to both *VS* and *S* membership functions, whereas 0.268 belongs more to *VS*, and 0.272 belongs more to *S* membership function.

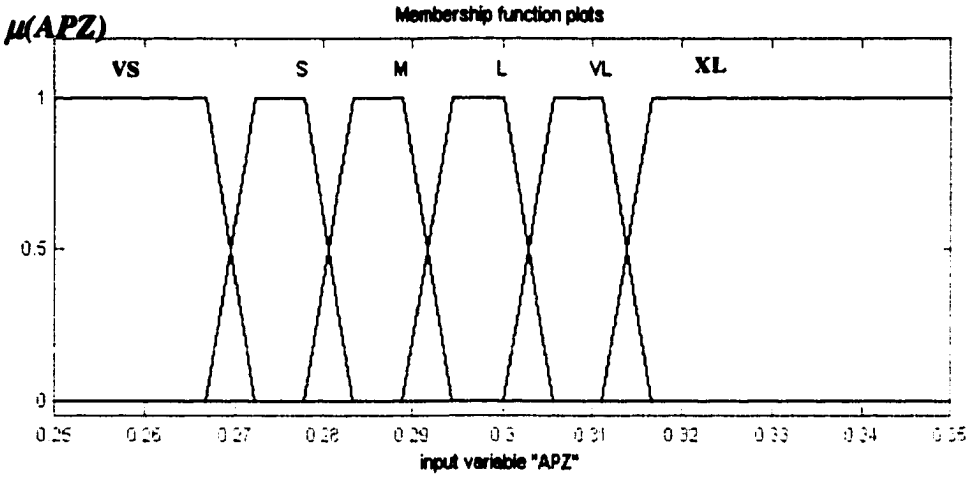


Figure 4.5 Membership functions for APZ

As a second example, Figure 4.6 illustrates the histogram of normalized PIB responses, which are concentrated mostly in the [0.65 0.93] range, the effective dynamic range of the PIB sensor. Therefore, the PIB MFs must be placed strategically in this interval to obtain the maximum discriminatory information from the sensor.

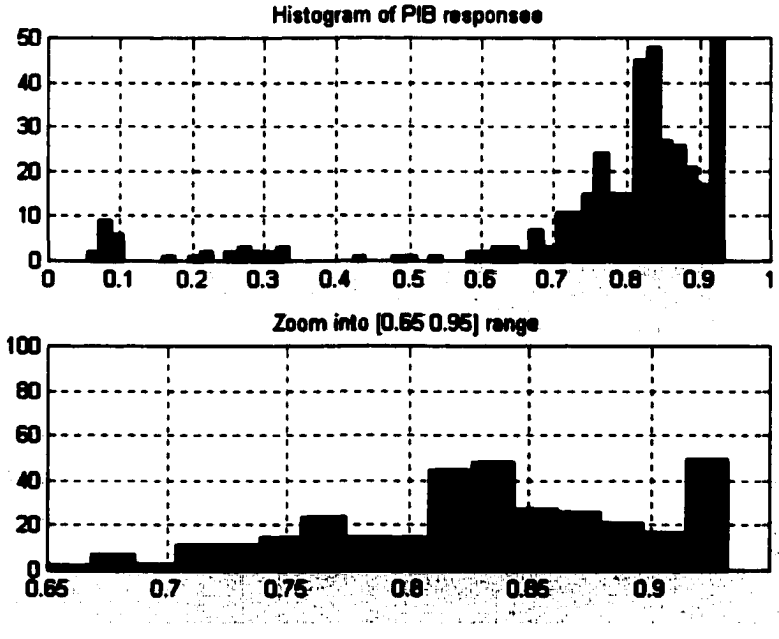


Figure 4.6 Histogram of PIB responses

Figure 4.7 illustrates the PIB membership functions placed on the basis of the histogram shown in Figure 4.6. All PIB responses less than 0.65 were assigned to the *VS* membership function (not shown in Figure 4.7). It is obvious from Figures 4.4 and 4.6 that a MF that is large for one sensor may be small for another. For example, the value 0.35 is a large value as an APZ response, but it represents a very small value for a PIB response, justifying the approach taken in the placement of the MFs.

Membership functions, determined similarly, for other sensor outputs were placed mainly in the [0.05 0.28] interval for DEGA, [0.15 0.22] interval for SG, [0.03 0.3] interval for OV275, and [0.12 0.5] interval for PDPP.

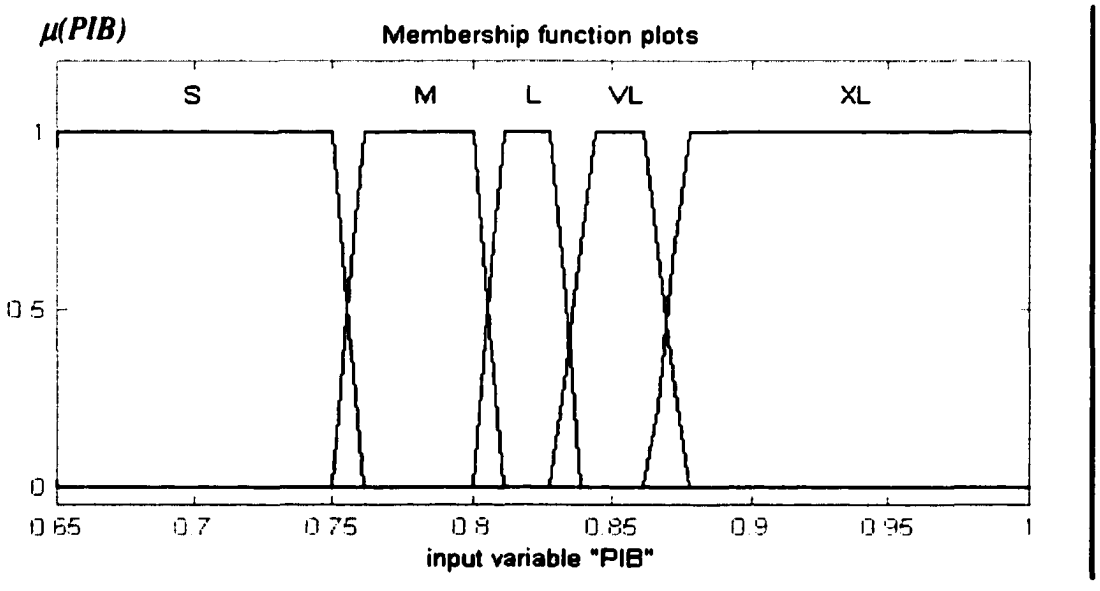


Figure 4.7 Membership functions for PIB

4.2.3 Rule Selection and Implication

The second step in designing an FIS is determining fuzzy rules, by which the FIS makes its classification decisions. Applying these rules to the data, and computing how much each rule *fires* is called the *implication* step in the fuzzy logic terminology. Fuzzy rules are usually of the following form:

IF <statement A> *AND / OR* <statement B> *AND / OR*.... <statement N> *THEN* <class = C>

The *IF* part is called the antecedent (or premise), and the *THEN* part is called the consequent (or conclusion) of the rule. The antecedent may involve a number of conditions joined by the logical operators AND or OR.

Several approaches can be used to determine the fuzzy rules. One of the commonly used methods relies on using past experience or expert advice, where the rules are determined by carefully examining and hand scoring the data. This method is particularly applicable to small databases. The 384-pattern database obtained for this study falls into this category. Other methods of determining rules include k-means or fuzzy c-means clustering algorithms [85, 87]. In these methods, the cluster centers are taken as rules, and the variances of the data from the cluster centers define the range of the rule.

The FNOSE fuzzy inference system employed the expert advice method for generating the fuzzy rule base, and 55 rules were extracted from data. The following example taken from the rulebase generated for this study illustrates the general form of the rules:

IF *(APZ is VL)* **AND** *(PIB is XL)* **AND** *(DEGA is VS)* **AND** *(SG is VS)* **AND** *(OV275 is VS)* **AND** *(PDPP is S)* **THEN** *(VOC1 is TCE) (0.5)*

This rule is interpreted by the FIS as follows: the dominant VOC (VOC1) is TCE, if the response of the APZ coated sensor is in the *VL* range, and the response of the PIB coated

sensor is in the *XL* range,..., and the response of the PDPP coated sensor is in the *S* range. The weight of this rule is 0.5. The weight of the rule determines how much importance should be given to rules that have identical antecedents, but different consequences. Recall that there are many similar patterns of different VOC classes, and these patterns generate rules with identical IF parts but different THEN parts. Depending on how often each rule of identical antecedents appears, a weight is assigned to that rule. Note that not all of the inputs may be necessary for each rule. A subset of inputs may be adequate to identify a certain class (please see the example below).

Rule selection is the heart of any FIS and must be done very carefully. Increasing the number of rules by adding new rules that cover only individual patterns may make the system too complicated and unstable, whereas having too few rules may not allow the system to generalize well. It is important to choose rules that would apply and correctly classify a large number patterns. It should be noted that making a rule for every case in the data will only make the system very unstable since there would be a lot of rules with the same antecedent (IF part), but different consequences (THEN parts).

Once the rules are selected and input values are fuzzified, the FIS then decides how much each rule is fired for a given input. Fuzzy rules which consist of statements combined through set-theoretic operations can be evaluated by computing the minimum of membership values for antecedents joined by AND, and computing the maximum of membership values for those joined by OR. In particular,

$$\Omega = A \cap B \cap ... \cap Z \Rightarrow \mu_{\Omega}(x) = \min(\mu_A(x), \mu_B(x), ..., \mu_Z(x)) = \mu_A(x) \wedge \mu_B(x) \wedge ... \mu_Z(x)$$
$$\Theta = A \cup B \cup ... \cup Z \Rightarrow \mu_{\Theta}(x) = \max(\mu_A(x), \mu_B(x), ..., \mu_Z(x)) = \mu_A(x) \vee \mu_B(x) \vee ... \mu_C(x)$$

4.2.4 Aggregation and Defuzzification

Due to fuzziness in the system, usually more than one rule fires for every set of inputs. The fuzzy output sets generated by the firing rules are then combined to determine the composite output. This procedure is called *aggregation*, and it is followed by the last step, *defuzzification*. A number of methods are available for both aggregation and defuzzification. Functions that are commonly used for defuzzification include centroid of area, mean of maximum, smallest of maximum, largest of maximum, bisector of area, and centroid of maximums [85]. Among these, centroid of area is used most often which is defined by Equation 4.2

$$Z_{CENTROID} = \frac{\int_z \mu_c'(z) \cdot z \cdot dz}{\int_z \mu_c'(z) \cdot dz} \tag{4.2}$$

where, μ_c' is the membership value of the aggregated output, and z is the independent variable. Figure 4.8 summarizes all the steps included on a sample input and sample fuzzy rules. Consider the input *APZ=0.282, PIB=0.875, DEGA=0.150, SG=0.250, OV274=0.30* and *PDPP=0.40*, and two sample fuzzy rules that fit these inputs:

IF *(APZ is M)* **AND** *(PIB is VL)* **AND** *(DEGA is M)* **THEN** *(VOC1 is OC)*

IF *(APZ is S)* **AND** *(PIB is XL)* **AND** *(DEGA is L)* **THEN** *(VOC1 is XL)*

Note that, as mentioned earlier, all inputs may not be necessary to identify a class, and the above two rules are examples of this case, where we have used only three inputs. In the following example, minimum of fuzzy membership values is used for implication, and summation of fuzzy output sets is used for aggregation, followed by centroid of area for defuzzification.

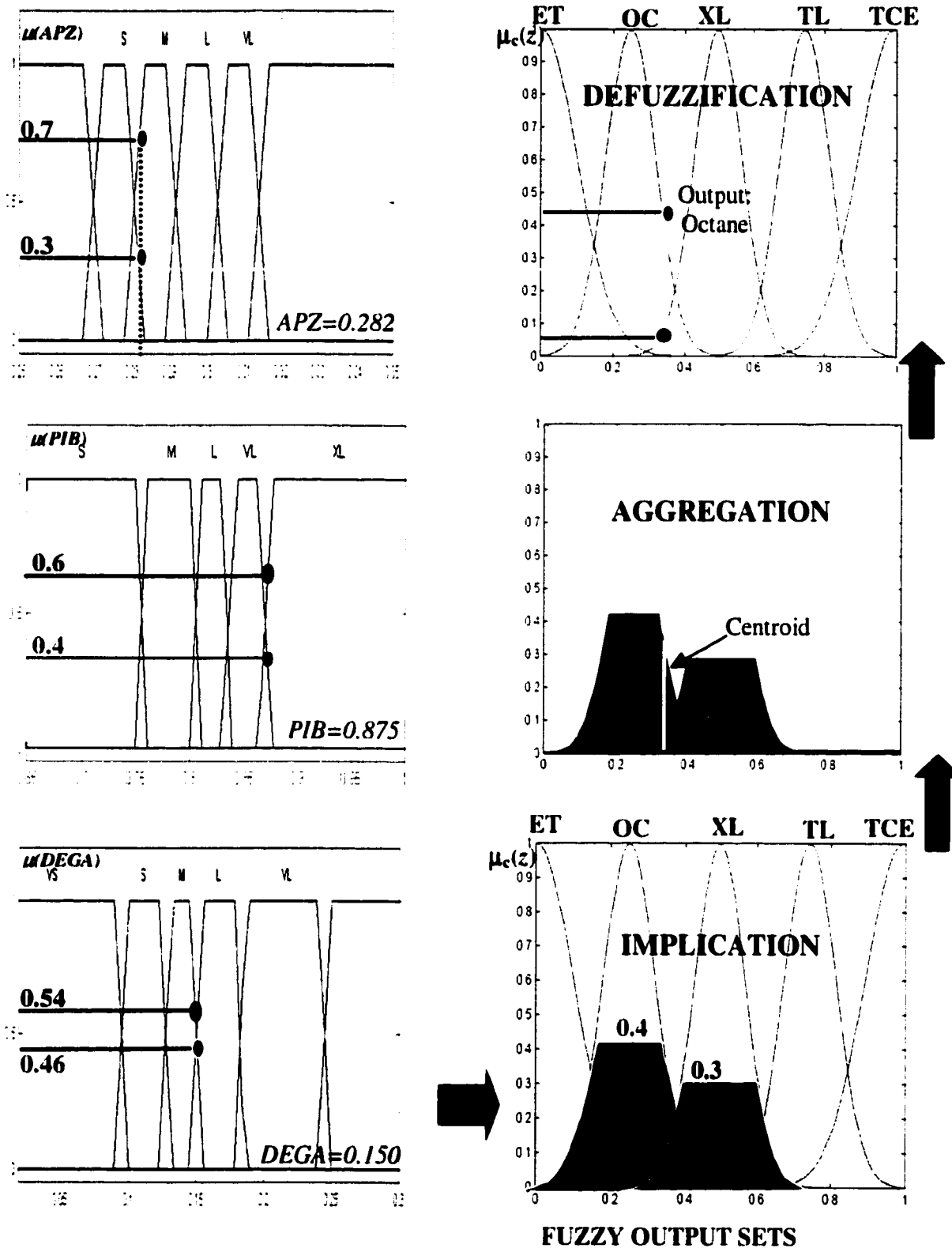


Figure 4.8 Sample FIS flow diagram

This defuzzification typically applies to *Mamdami Type* fuzzy inference systems, for which the outputs are also defined by membership functions. Mamdami type systems become particularly efficient when the individual classes might be related to each other, or when a particular input may belong to more than one class with varying membership values. In systems, where the output classes are independent, or where each input may belong to one and only one class, a second type of FIS becomes more useful. *Sugeno Type* fuzzy systems, such as FNOSE, use crisp values, rather than fuzzy membership functions for outputs. This automatically eliminates the defuzzification step, since defuzzification becomes simply a weighted average of how much each rule fires for every input. The crisp outputs were given numerical values such as TL=0, XL=0.25, TCE=0.5, OC=0.75, ET=1 for the five dominant VOCs. Since any input can (and frequently does) fire more than one rule (due to overlap in membership functions), the output is typically an intermediate value of those listed above. Therefore, the output ranges were determined as follows:

0 ~ 0.125 : TL

0.125 ~ 0.375 : XL

0.375 ~ 0.625: TCE

0.625 ~ 0.875: OC

0.875 ~ 1 : ET.

In some cases, the output was at the border of the output ranges listed above. The performance of this system is presented in the next section, where the classification percentages are given as intervals. The lower limits of the intervals assume all border cases as failure, and the upper limits assume all border cases as success.

4.2.5 Results for the Neurofuzzy Approach

4.2.5.1 First Stage: Performance for Dominant VOC Identification

Fifty-five rules were manually chosen from the tables given Appendix II. These tables were generated by a program, which converted all signals with numeric values to linguistic values depending on the sensor and its individual membership functions. The 55-rule rule base given in Appendix III was obtained by hand scoring the data. It should be noted that these rules did not cover the entire database, that is, there were patterns that were not covered by any of the rules which contributed to the error of the system. Obviously, including these patterns in the rule base would make the system classify those correctly, but including rules that only apply to single cases causes the system to memorize certain patterns and prevents it from learning and generalizing. Table 4.1 summarizes the FNOSE performance.

4.2.5.2 Second Stage: Identification of Secondary VOCs

Although a NN was unable to classify signals according to their dominant VOCs, it was possible to train a NN to recognize the secondary VOCs, once the dominant one was

Table 4.1 Performance of FNOSE for dominant VOC identification

Mixtures of	Total number of patterns	Worst Best performance	Average Percentage	Remarks Misclassified ones were classified as
ETHANOL	64	55/57	87%	TCE
TOLUENE	80	70/70	87%	TCE(3), ET(3), XL(2)
XYLENE	80	71/77	92%	TOLUENE
OCTANE	80	71/73	90%	TCE
TCE	80	70/70	87%	XL(8), ET, OC
TOTAL	384	336/347	89%	Mostly TCE and XL

identified using FNOSE. Identification of secondary VOCs was therefore performed by one of five MLPs, each specifically trained to recognize the mixtures of one dominant VOC. Table 4.2 presents the individual network structures and the number of training data used for each network, whereas Figure 4.9 illustrates the architecture of a 6x20x5 MLP.

Table 4.2 Secondary VOC identification network characteristics

Network Designed For the mixtures of	Network Architecture	Error Goal	Number of training data	Total number of signals
ETHANOL	6x20x5	0.05	30	64
TOLUENE	5x30x5	0.05	40	80
XYLENE	5x24x5	1.20	40	80
OCTANE	5x20x5	0.05	40	80
TCE	5x30x5	0.05	40	80

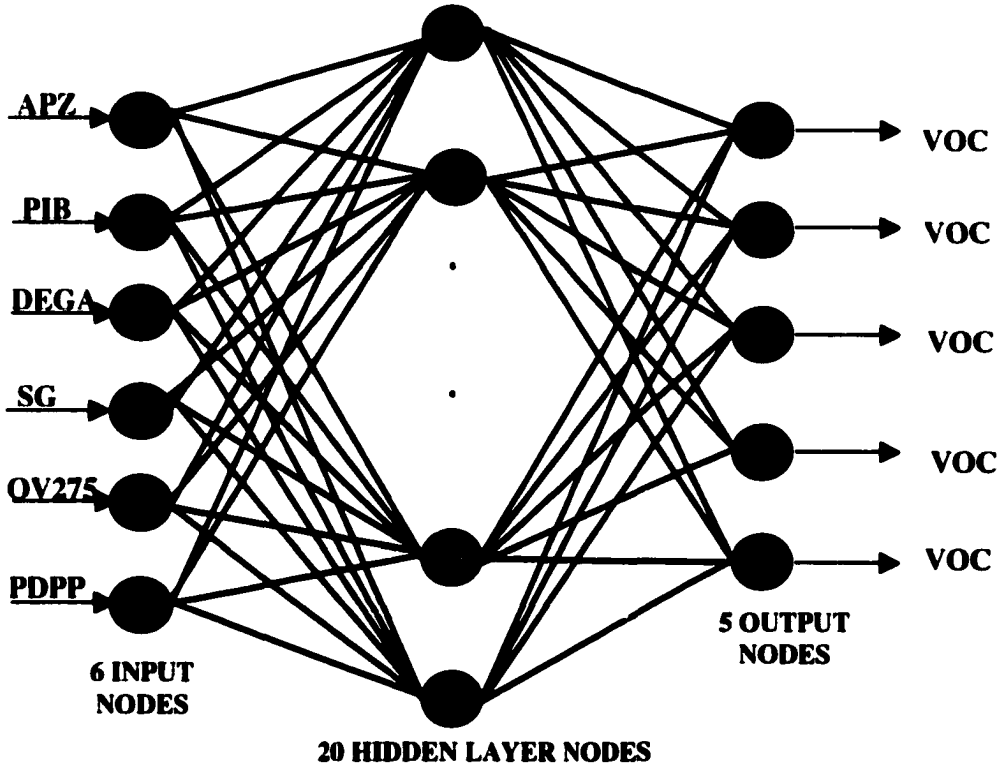


Figure 4.9 MLP architecture

4.2.5.3 Results and Discussion of Second Stage Performance

The results of secondary VOC networks are summarized in Table 4.3. The number that is given next to each mixture refers to the number of misclassified patterns for that mixture, out of sixteen. There were 26 misclassifications out of 384 signals, giving a classification performance of 93% over the entire database, or 87% over the test database.

Table 4.3 Performance of the secondary VOC neural networks

TCE Mixtures		TL Mixtures		XL Mixtures		OC Mixtures		ET Mixtures	
TCE & TL	0	TL & ACN	1	XL & ACN	3	OC & ACN	2	ET & ACN	0
TCE & MEK	0	TL & MEK	0	XL & ET	4	OC & MEK	0	ET & MEK	0
TCE & TCA	0	TL & HX	2	XL & HX	2	OC & TL	1	ET & HX	0
TCE & HX	0	TL & ET	1	XL & MEK	1	OC & ET	3	ET & TCA	0
TCE & ET	0	TL & TCA	1	XL & TCA	4	OC & TCA	1		

It is interesting to note from Tables 4.2 and 4.3 that xylene mixtures were the most difficult ones to identify. As Table 4.2 points out, the lowest error goal (mean square error) that was reached by any network training for xylene mixtures was 1.2, whereas the lowest achieved error goal for networks of other mixtures was 0.05. Table 4.3 shows the effect of a higher error goal in network training, because xylene mixtures had the largest number of misclassification. These results agree with our previous knowledge of xylene mixtures. Recall that responses of all sensors to xylene were significantly larger compared to the responses to other VOCs, and this made the detection and identification of other VOCs very difficult in the presence of xylene.

4.2.6 Overall Performance

The process described above is a two-stage scheme, where the dominant VOC is first identified using an FIS, and the secondary VOC is then identified using a MLP. The overall performance depends on the combined performance of both systems. In particular, the identification of the secondary VOC is irrelevant, if the dominant VOC identification is incorrect. The actual classification performance over the entire database, is therefore 83%, which is 89% (first stage) of 93% (second stage). Considering that no preprocessing was performed, this can be viewed as a very promising performance. The FNOSE / ANN system can be summarized by the block diagram in Figure 4.10.

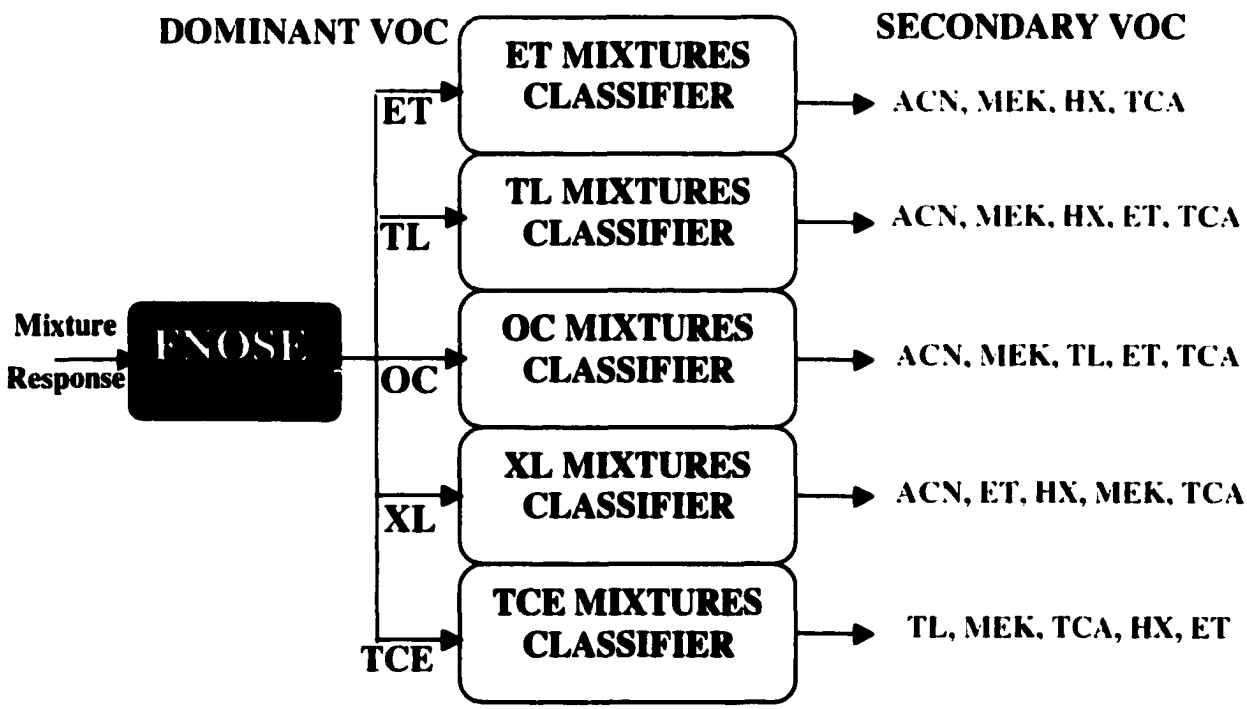


Figure 4.10 VOC Mixture identification system using FNOSE and a neural network

4.3 Feature Range Stretching (FRS) for Enhancing Pattern Separability

4.3.1 Approach

Of the two stages described in the neurofuzzy approach utilizing an FIS for dominant VOC identification and an ANN for secondary VOC identification, the first stage is the more challenging one. This is due to strong similarities between signals of different dominant VOCs. A new preprocessing scheme was developed in an attempt to increase the intercluster distances between the signals, so that both dominant and secondary VOCs could be identified by neural networks.

Our initial approach was based on the use of a companding (compress/expand) function to enhance the subtle differences between similar responses. Companding is a standard nonlinear scaling procedure used in communication systems to amplify minor differences in signals [103]. Using a companding function as a preprocessor prior to training allowed neural networks to achieve a reasonable overall classification performance (86% overall classification, compared to 83% of FNOSE+ANN). However, a new method for stretching dynamic range of features was developed and adopted for improved performance and robustness of the overall classification system.

This preprocessing scheme was inspired by the membership function selection scheme that was used in FNOSE, the companding scheme mentioned above, and the histogram equalization technique used in image processing for improving image quality. In effect, this preprocessing algorithm maps a narrow range of sensor responses to a wider range to increase the separability of the data.

For example, as seen in Figure 4.4, all normalized APZ responses were between 0.25 and 0.32. This range can be mapped to [0 1] by using a suitable function. Similarly, all normalized PIB responses, which were predominantly in the [0.65 0.95] interval, can also be mapped to [0 1] range by using another suitable function, and so forth. In effect, the ranges of features can be stretched from a narrow region to the full range of [0 1], hence, *feature range stretching (FRS)*. This method can be thought of as a one-dimensional version of histogram equalization used in image processing for improving image quality, and we can follow a similar procedure to that given in [104] to find the suitable mapping function. Just like in histogram equalization, where the problem is to map the gray levels from a narrow range to a full dynamic range, our problem is to find a transformation function of the form

$$y = T(x) \tag{4.3}$$

which will map a given narrow sub interval of [0 1] to the full range. Intuitively, it is reasonable to expect the following properties from such a mapping function:

i. $T(x)$ should be a single valued and monotonically increasing function in the interval [0 1], so that the function preserves the relative amplitude ordering within the input signal, and

ii. $T(x)$ should satisfy $0 \le T(x) \le 1$ for $0 \le x \le 1$, so that the mapping stays within the interval [0 1].

The values of any specific sensor output can be considered as random quantities in [0 1] interval. If we further assume that these random quantities are continuous (discrete case is a natural extension to this, as discussed later), the original values, x, and the transformed values y, can be represented by their probability density functions, $p_x(x)$ and $p_y(y)$, respectively.

From the theorem of transformation on random variables, if $p_x(x)$ and $T(x)$ are known, we can compute $p_y(y)$ by

$$p_y(y) = p_x(x) \cdot \left. \frac{dx}{dy} \right|_{x=T^{-1}(y)} \tag{4.4}$$

provided that $T^{-1}(y)$ is also monotonically increasing function [105].

Now consider $f_x(x)$, the cumulative distribution function of x, for which the conditions listed above are satisfied:

$$f_x(x) = \int_0^x p_x(\varphi) d\varphi \tag{4.5}$$

If we use this cumulative distribution function as our transformation function, we obtain

$$y = T(x) = f_x(x) = \int_0^x p_x(\varphi) d\varphi \tag{4.6}$$

from which we can compute the derivative dy/dx as

$$\frac{dy}{dx} = \frac{d}{dx}\left(\int_0^x p_x(\varphi) d\varphi \right) = p_x(x) \tag{4.7}$$

Substituting Equation 4.7 into Equation 4.4, we obtain

$$\begin{aligned} p_y(y) &= p_x(x) \cdot \left. \frac{dx}{dy} \right|_{x=T^{-1}(y)} \\ &= p_x(x) \cdot \left. \frac{1}{p_x(x)} \right|_{x=T^{-1}(y)} \\ &= 1 \qquad \text{for } 0 \le y \le 1 \end{aligned} \tag{4.8}$$

which is a uniform density function, regardless of $T^{-1}(y)$. In other words, if we use the cumulative distribution functions as our mapping function, the transformed values will be uni-

formly distributed in the [0 1] interval. Consequently, values squeezed within a narrow subinterval of [0 1] are stretched to the [0 1] interval.

Note however, that the patterns that are of interest are discrete, and therefore the above arguments need to be modified for the discrete case. In discrete case, the cumulative distribution function can be approximated by a running sum of the form

$$y[n] = T(y_n) = \sum_{i=0}^{n} \frac{\psi_i}{\Psi} = \sum_{i=0}^{n} p_r(x_n) \tag{4.9}$$

where ψ_i is the frequency of i^{th} sample, and Ψ is total number of samples (384 for the VOC database).

Due to the nature of the patterns that are of interest in this study, it is not very easy to compute the above given distributions, since it requires the computation of the frequency of each value. Considering that this procedure needs to be repeated for each sensor, the computation of the exact transformation function(s) becomes a formidable task. Fortunately, it can easily be approximated, as discussed below.

Figure 4.11 shows a different way of interpreting the histogram of APZ values. Instead of plotting the frequency of occurrence of values (as in Figure 4.4), all 384 values are plotted in an ascending order against an index running from 1 to 384. As seen in this figure, a large number of APZ responses fall into the very close vicinity of 0.3. We would like to transform this characteristic such that all values from zero to one can be (approximately) equally utilized. Recall that the exact transformation function that is required is the cumulative distribution function of x, which in this case is the APZ response. This cumulative distribution function, $f_{APZ}(x)$, can be approximated simply by inverting the curve in Figure 4.11 and plotting it against the interval [0 1] divided into 384 equal partitions.

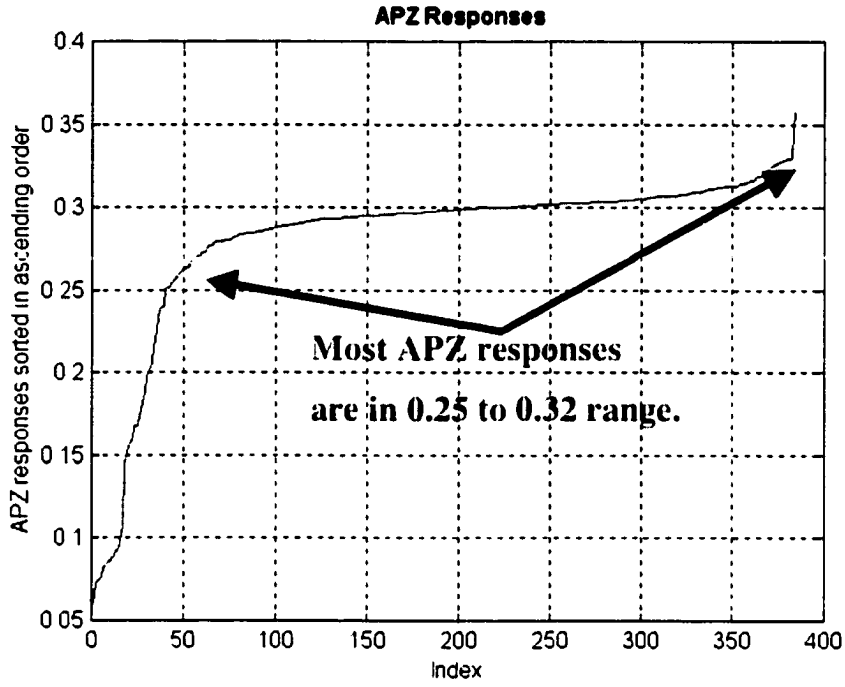


Figure 4.11 APZ responses sorted in ascending order

Figure 4.12 illustrates this point, where the *horizontal* axis is the sorted APZ responses, and the vertical axis is the desired [0 1] range. Also shown in Figure 4.12 is a smoother charactersitic that is superimposed on $f_{APZ}(x)$. This is an approximation of the original characteristic, obtained by a sigmoid function centered at 0.27 and multiplied by 50. The approximation function was used for the actual mapping due to its simplicity and robustness to noise.

$$y(x) = 50 \cdot \frac{1}{1+e^{-(x-0.27)}} \tag{4.10}$$

This function was then realized by using a radial basis function neural network (RBFNN) which is commonly used for function approximation and realization.

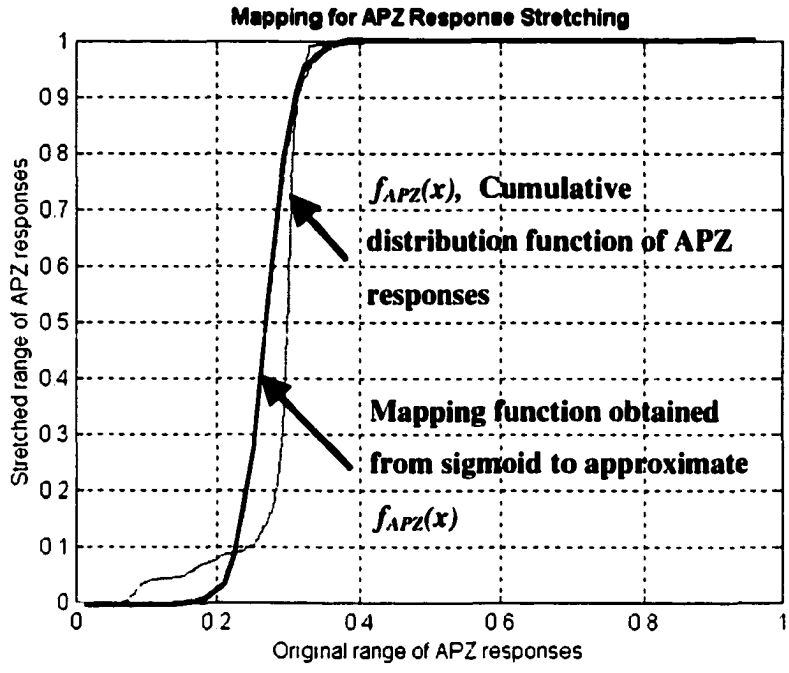


Figure 4.12 Stretching APZ responses

Figure 4.13 illustrates the sorted PIB responses, whereas Figure 4.14 shows the cumulative distribution function obtained by inverting the curve in Figure 4.13 and plotting it against the desired range of [0 1]. The smoother characteristic is the actual mapping function obtained by the sigmoid.

$$y(x) = 20 \cdot \frac{1}{1+e^{-(x-0.82)}} \tag{4.11}$$

As a last example, Figure 4.15 illustrates sorted PDPP responses, from which we see that most PDPP responses are in the 0.15 to 0.5 range. Figure 4.16 shows the mapping function, $f_{PDPP}(x)$, obtained by inverting the above characteristic and plotting it against the desired range of [0 1]. Note that in this case, a linear combination of two sigmoid functions is used to obtain the overall desired characteristic.

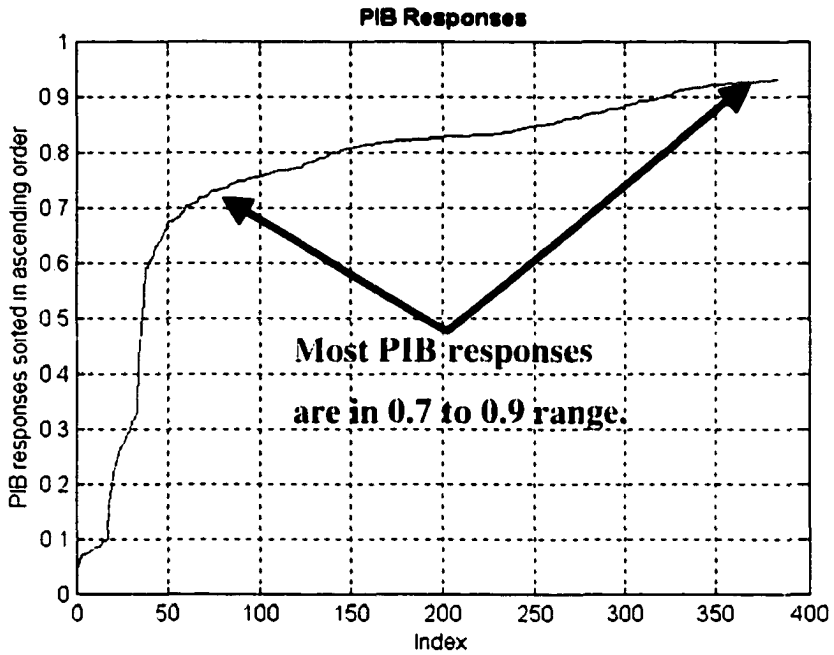


Figure 4.13 PIB responses sorted in ascending order

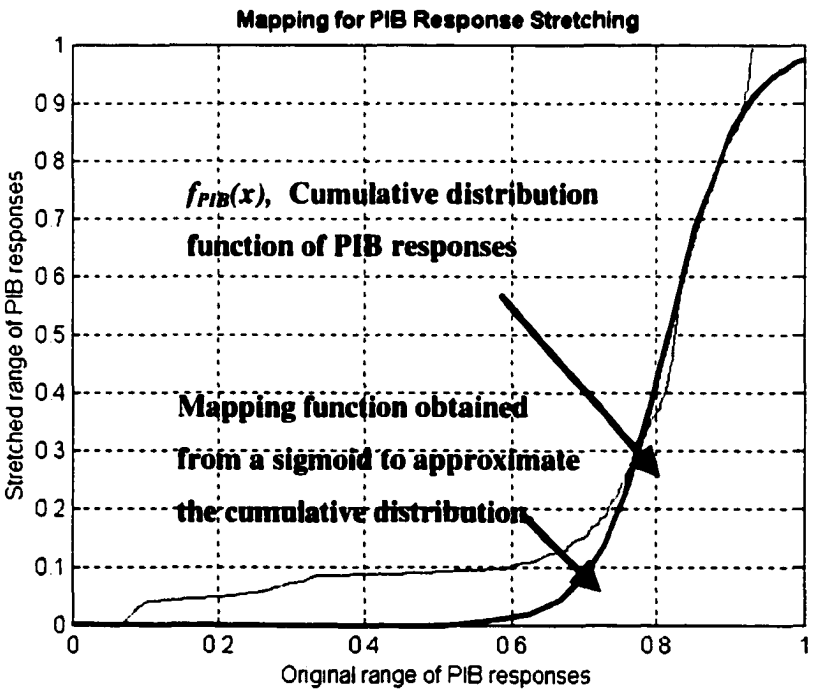


Figure 4.14 Stretching PIB responses

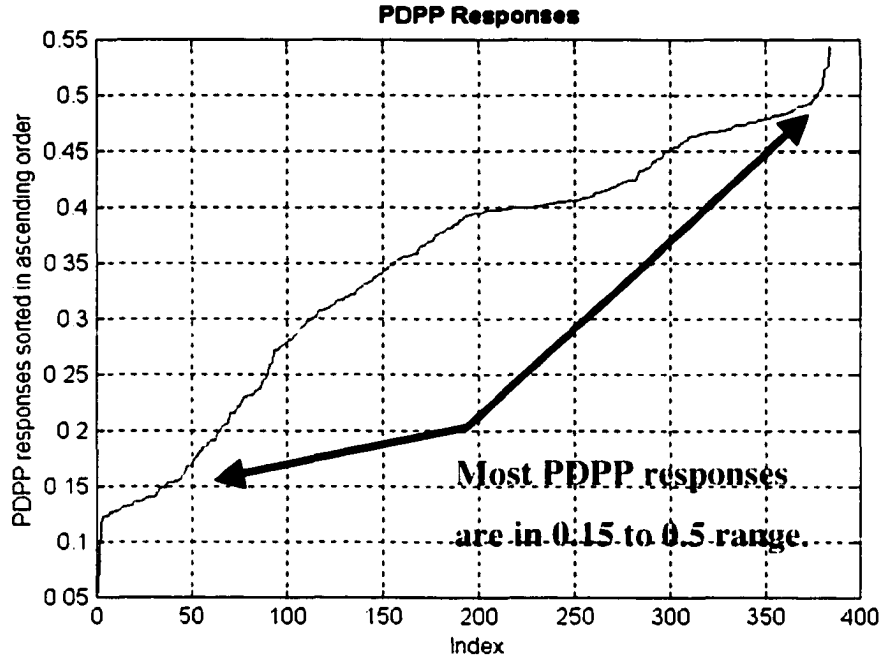


Figure 4.15 PDPP responses sorted in ascending order

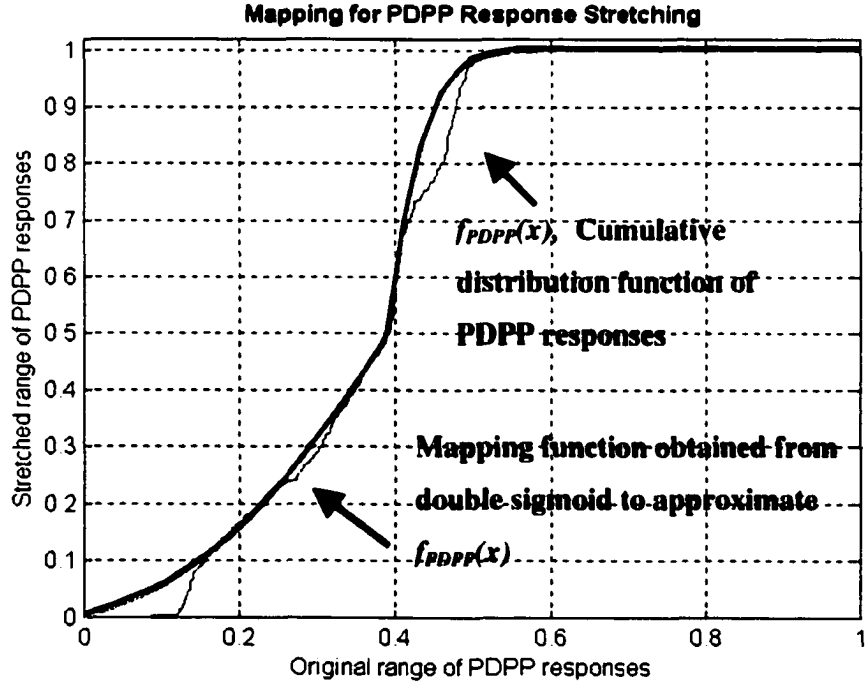


Figure 4.16 Stretching PDPP responses

Also shown in Figure 4.16 is the approximated mapping function, obtained from the two sigmoid functions as shown below.

$$y_1(x) = 8 \cdot \frac{1}{1+e^{-(x-0.37)}} - 0.05$$

$$y_2(x) = 35 \cdot \frac{1}{1+e^{-(x-0.39)}} \tag{4.12}$$

$$y(x) = y_1(x)\big|_{x=0\sim0.39} + y_2(x)\big|_{x=0.39\sim1}$$

Mapping functions were generated for the other three sensors in a similar fashion.

4.3.2 Identification of Dominant and Secondary VOCs using FRS

Similar to the earlier approach, a two-stage procedure was implemented to identify dominant and secondary VOCs. However, all signals were preprocessed by normalization followed by FRS prior to network training. The first network identified the dominant VOC in the mixture. Depending on the outcome of this network, one of the five secondary VOC networks was used, each of which was trained to recognize the secondary VOCs in the presence of a specific dominant VOC.

A two hidden layer multilayer perceptron of 6x50x20x5 architecture was trained using backpropagation. The randomly selected training data consisted of 153 patterns uniformly chosen from mixtures of five dominant VOCs.

4.3.3 Results for FRS Processed VOC Identification

Table 4.4 presents the results of the dominant VOC neural network. Numbers given next to mixture names are the number of misclassifications out of 16 different concentration combinations of two VOCs.

Table 4.4 Performance of dominant VOC network, trained with FRS processed data

TCE Mixtures	TL Mixtures	XL Mixtures	OC Mixtures	ET Mixtures
TCE & TL 5	TL & ACN 0	XL & ACN 0	OC & ACN 0	ET & ACN 0
TCE & MEK 1	TL & MEK 0	XL & ET 0	OC & MEK 0	ET & MEK 0
TCE & TCA 0	TL & HX 5	XL & HX 0	OC & TL 3	ET & HX 0
TCE & HX 0	TL & ET 0	XL & MEK 0	OC & ET 0	ET & TCA 0
TCE & ET 1	TL & TCA 0	XL & TCA 0	OC & TCA 0	

The total number of misclassifications was 15, giving a classification performance of 94% over the training data, and 96% over the entire database. Once the dominant VOC was identified, the appropriate secondary VOC network was used to identify the secondary VOC. The architectures of these networks are given in Table 4.5.

Table 4.5 Secondary VOC identification network characteristics

Network Designed For the mixtures of	Network Architecture	Error Goal	# of training data	Total # signals for this mixture
ETHANOL	6x20x4	0.05	24	64
TOLUENE	6x20x7	0.01	41	112
XYLENE	6x30x5	0.05	40	80
OCTANE	6x20x4	0.05	24	64
TCE	6x30x4	0.05	24	64

The performances of the secondary VOC networks are given in Table 4.6. The total number of misclassifications in this case was 16, which was superior to earlier results. Only 153 signals were used in the training database, as opposed to 190 used with the earlier secondary VOC networks of neurofuzzy approach. The total classification performance over the test data was 93% and over the entire database, it was 96%.

Table 4.6 Performance of the secondary VOC network, processed with FRS

TCE Mixtures		TL Mixtures		XL Mixtures		OC Mixtures		ET Mixtures	
TCE & TL	0	TL & ACN	0	XL & ACN	0	OC & CAN	0	ET & ACN	0
TCE & MEK	0	TL & MEK	2	XL & ET	2	OC & MEK	0	ET & MEK	1
TCE & TCA	0	TL & HX	3	XL & HX	3	OC & TL	0	ET & HX	0
TCE & HX	0	TL & ET	0	XL & MEK	1	OC & ET	3	ET & TCA	0
TCE & ET	0	TL & TCA	0	XL & TCA	1	OC & TCA	0		

The mixtures of toluene and xylene with secondary VOCs had the largest number of misclassifications for the secondary VOC. This is not surprising since these two VOCs exhibit very large responses in all sensors, masking the contribution of the secondary VOC. It is also worth mentioning that the xylene network was able to converge to an error minimum of 0.05 (see Table 4.5), whereas the xylene network used in the neurofuzzy approach that was trained with data without FRS preprocessing was not able to converge to an error goal smaller than 1.2 (see Table 4.2).

As it was in the FNOSE case, the overall performance of the two-stage identification system depends on the individual performances of both stages. In particular, the performance of the second stage is meaningless, if the dominant VOC is identified incorrectly. Therefore, the overall classification performance over the entire database was 92% (96% of 96%).

Considering the smaller training database size, small feature vector size, and the network error minima achieved along with the results obtained, FRS preprocessing not only improved the overall system performance, but also made it more robust.

4.3.4 Principal Component Analysis (PCA)

The effect of FRS processing was also investigated by using the principal component analysis (PCA), a well-known dimensionality reduction and classification algorithm. PCA was performed on raw data, as well as on FRS processed data. Three principal components, corresponding to the largest three eigenvalues of the covariance matrix (of the data), were plotted on a 3D plot to illustrate the effect of these processing schemes on intercluster and intracluster distances.

Principal component analysis (PCA) is commonly used to reduce the dimensionality of feature vectors [90]. The idea is to find a set of n orthogonal vectors along which the m dimensional data has the largest variance. Large variance is usually interpreted as *more information* and therefore, the m dimensional feature vector is replaced by its n dimensional projection on these orthogonal vectors, where $n<m$.

Principal components are computed by projecting the data on the orthogonal vectors, which are eigenvectors of the covariance matrix, $\mathbf{C} = E\{\mathbf{x} \cdot \mathbf{x}^T\}$. E is the expected value operator, $\mathbf{x}$ is the m dimensional feature vector, and $\mathbf{x}^T$ is the transpose of $\mathbf{x}$. The eigenvectors corresponding to the n largest eigenvalues are selected, and the principal components are then computed by projecting the data onto n dimensional space spanned by the selected eigenvectors.

It is often possible to reduce the dimensionality of the feature vector down to two or three, which allows simple visual classification of the patterns when their principal components are plotted on a two or three-dimensional plot. In the following paragraphs, these plots are used to illustrate the effect of the preprocessing algorithms discussed above. Detailed de-

scription of PCA is available in most neural network and pattern recognition texts, such as [90].

PCA was first applied to unprocessed, normalized raw data to reduce the dimensionality from six to three, and Figure 4.17 illustrates the data clustered into five clusters in a 3D space corresponding to the five dominant VOCs. Note that, as expected from the sensor frequency responses, OC and XL responses overlap considerably, and TL responses are very close to OC and XL responses.

The PCA of the FRS processed data was then computed as shown in Figure 4.18. As seen in this figure, FRS processing considerably increased the intercluster distances. All mixtures are now clearly separable on the basis of their dominant VOCs. This figure clearly demonstrates the viability of the FRS preprocessing approach.

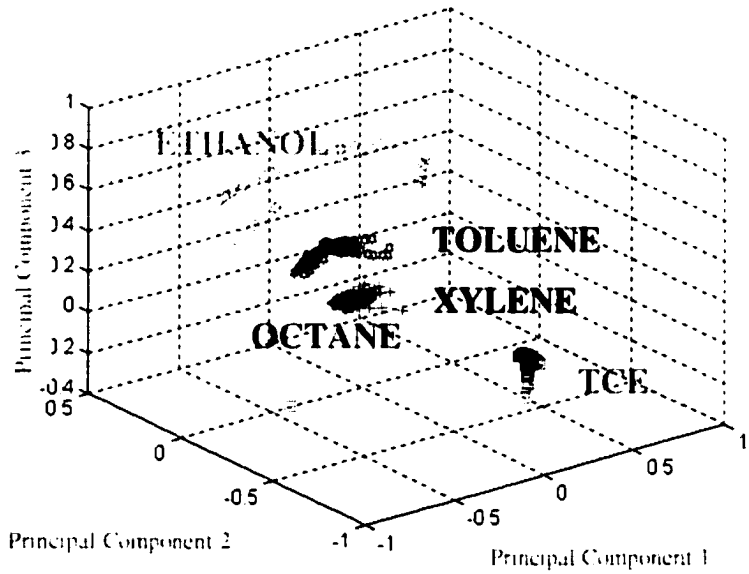


Figure 4.17 Principal components of raw data

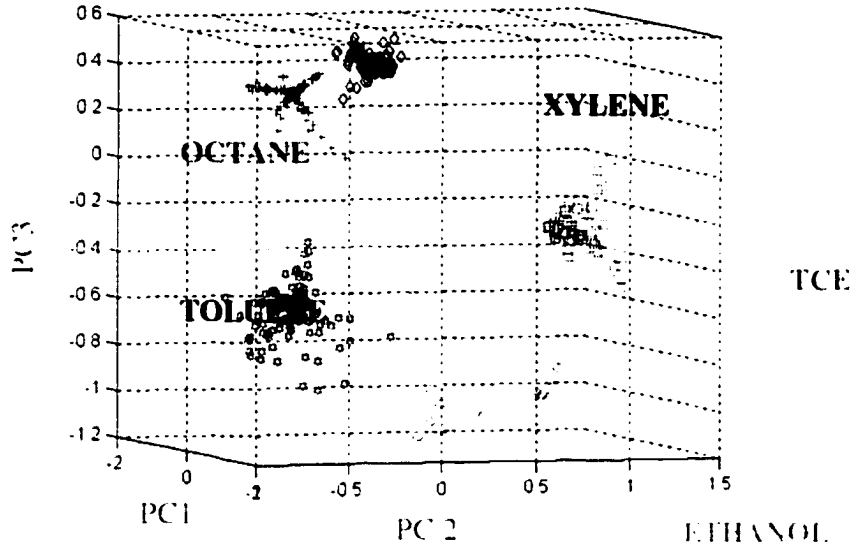


Figure 4.18 Principal components of FRS processed vectors

4.4 Nonlinear Cluster Transformation for Enhancing Pattern Separability

4.4.1 Motivation

The FRS preprocessing introduced in the previous section exclusively targets (and achieves) increasing the intercluster distances, however, it also increases the intracluster distances as well. Furthermore, it requires a nonlinear stretching function to be determined manually for every feature.

One well-known method specifically designed to increase the intercluster distances and reduce the intracluster distances is Fisher's linear discriminant (FLD) analysis. However, the original motivation behind the FLD was reducing the dimensionality of the data subject to maximizing the intercluster distances and minimizing the intracluster distances. As a conse-

quence of this, FLD has serious limitations regarding the dimensionality of the data, number of classes and number of samples within the data.

In this section, these limitations of FLD are discussed, and an intuitive cluster transformation method is proposed for increasing the intercluster distances while keeping the intracluster distances constant. In *nonlinear cluster transformation (NCT)*, a training dataset is used to translate the cluster centers away from each other and a generalized regression neural network (GRNN) is employed to learn the functional mapping between original clusters and transformed clusters. The performance of this proposed method is tested on two synthetic benchmark databases of low separability as well as the VOC mixture database. Initial results obtained using NCT have been very promising, demonstrating the effectiveness of the approach in separating patterns that have small intercluster distances.

4.4.2 Background

The required separability for challenging pattern recognition problems is often obtained by using appropriate feature extraction algorithms as a preprocessing step to classification. The fundamental objective of feature extraction is to reduce the dimensionality of pattern vectors without losing discriminatory information. The general problem of feature extraction can be formulated as one of determining a mapping of the form $\mathbf{y} = f(\mathbf{x})$, or $\mathbf{y} = \mathbf{W}^T\mathbf{x}$, that transforms pattern vectors onto a lower dimensional feature space in which the corresponding feature vectors are separable. The Fisher Linear Discriminant (FLD) was one of the first methods proposed to achieve dimensionality reduction, based on maximizing the ratio of intercluster to intracluster distances. The FLD algorithm projects the data onto a lower dimensional space where this criterion function is maximized. Consequently, FLD is a feature re-

duction algorithm that ensures maximum separability of patterns in the transformed space. However, as discussed in the following paragraphs, dimensionality reduction may not always be very beneficial for increasing pattern separability. In fact, the drastic dimensionality reduction that FLD provides imposes rather stringent limitations on the data with respect to the number of patterns in the training database, number of classes, and the dimensionality of patterns.

4.4.3 Fisher's Linear Discriminant and Its Limitations

FLD has enjoyed much attention and success largely as a technique for reducing the dimensionality of a classification problem by projecting the data instances onto a new space of lower dimension.

Consider a multi-class classification problem and let C be the number of classes. For the i^{th} class, let $\{\mathbf{X}_i\}$ be the set of patterns in this class, $\mathbf{m}_i$ be the mean of vectors $\mathbf{x} \in \{\mathbf{X}_i\}$, n_i be the number of patterns in $\{\mathbf{X}_i\}$. Let $\mathbf{m}$ be the mean of all patterns in all C classes. Then the within scatter matrix $\mathbf{S_W}$, and between scatter matrix $\mathbf{S_B}$ are defined as follows:

$$\begin{aligned} \mathbf{S}_W &= \sum_{i=1}^{C} \sum_{\mathbf{x} \in X_i} (\mathbf{x} - \mathbf{m}_i) \cdot (\mathbf{x} - \mathbf{m}_i)^T \\ \mathbf{S}_B &= \sum_{i=1}^{C} n_i (\mathbf{m} - \mathbf{m}_i) \cdot (\mathbf{m} - \mathbf{m}_i)^T \end{aligned} \tag{4.13}$$

Note that $\mathbf{S_W}$ is a measure of the intracluster distances, and $\mathbf{S_B}$ is a measure of the intercluster distances. The transformation, which is also the projection from the original feature space onto a lower dimensional feature space, can be expressed as

$$\mathbf{y} = \mathbf{W}^T \cdot \mathbf{x} \tag{4.14}$$

where the column vector **y** is the feature vector in the projected space corresponding to pattern **x**. The optimum matrix **W** is obtained by maximizing the criterion function

$$J(\mathbf{W}) = \mathbf{S}_{BF} \Big/ \mathbf{S}_{WF} \tag{4.15}$$

where $\mathbf{S}_{BF}$ and $\mathbf{S}_{WF}$ are the corresponding scatter matrices in the (feature) projection space. It can be shown that $\mathbf{S}_{BF}$ and $\mathbf{S}_{WF}$ can be written as [94]

$$\begin{aligned} \mathbf{S}_{BF} &= \mathbf{W}^T \mathbf{S}_B \mathbf{W} \\ \mathbf{S}_{WF} &= \mathbf{W}^T \mathbf{S}_W \mathbf{W} \end{aligned} \tag{4.16}$$

Therefore, the criterion function can be represented in terms of the scatter matrices of the original patterns.

$$J(\mathbf{W}) = \frac{\mathbf{W}^T \mathbf{S}_B \mathbf{W}}{\mathbf{W}^T \mathbf{S}_W \mathbf{W}} \tag{4.17}$$

$J(\mathbf{W})$ is a vector valued function, and the determinant of this function can be used as a scalar measure of $J(\mathbf{W})$. The columns of **W** that maximize the determinant of $J(\mathbf{W})$ are then the eigenvectors that correspond to the largest eigenvalues in the generalized eigenvalue equation [94]

$$\mathbf{S}_B \mathbf{w}_i = \lambda_i \mathbf{S}_W \mathbf{w}_i \tag{4.18}$$

For nonsingular $\mathbf{S}_W$, Equation 4.18 can be written as

$$\mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{w}_i = \lambda \mathbf{w}_i \tag{4.19}$$

From Equation 4.19, we can directly compute the eigenvalues λ_i and the eigenvectors $\mathbf{w}_i$, which then constitute the columns of the **W** matrix.

The limitations of the FLD method can be explained with the help of two theorems, the proofs of which are straightforward consequences of linear algebra theory.

Theorem 1. *Regardless of the dimension of the original pattern, the FLD transforms a pattern vector onto a feature vector, whose dimension can be at most C-1, where C is the number of classes.*

Proof: The rank of a matrix that is obtained by multiplying a vector by its transpose is always one. Also, note from Equation 4.13 that the between class scatter matrix $\mathbf{S_B}$ is obtained by adding C rank 1 matrices, only $C-1$ of which are linearly independent. The rank of $\mathbf{S_B}$ can therefore be at most $C-1$. Consequently, only $C-1$ eigenvalues can be non-zero, and $\mathbf{W}$ is then obtained by taking the eigenvectors $\mathbf{w_i}$ that correspond to nonzero eigenvalues. If the original data is d dimensional, then the $\mathbf{W}$ matrix will be of size $[d \times C-1]$, and when applied to the original data, this projection will reduce the dimensionality to $C-1$. ◆

In most signal processing and pattern recognition applications, the dimensionality of the original data is significantly larger than number of classes, and therefore FLD provides a significant dimensionality reduction for these cases. However, not all applications possess this property, and for those applications, FLD cannot be used. In fact, the gas-sensing problem discussed later in this paper is an example of such an application. Furthermore, the dimension of the new space is predetermined by the number of classes, and cannot be changed. This means that, regardless of the dimensionality of the original data, the reduced dimension will always be $C-1$. In certain databases where C is in the order of 2-5 and where d is in the order of 100~1000, the reduced dimensionality with only $C-1$ features may not be adequate to distinguish the classes. In other words, only a few features may not be able to provide the necessary information to classify instances of such a database.

Theorem 2. *The matrix $\mathbf{S_W}$ is nonsingular only if $N-C < d$, where N is the number of training data and d is the dimension of the pattern vector.*

Proof: The solution to the generalized eigenvalue problem of Equation 4.19 is based on the assumption of $\mathbf{S_W}$ being nonsingular. Note that the inside summation of $\mathbf{S_W}$ in Equation 4.13 adds n_i matrices, $(n_i - 1)$ of which are linearly independent. Therefore, the rank of this inner sum can at most be $(n_i - 1)$. The outer sum of $\mathbf{S_W}$ adds C such matrices, where C is the number of classes. The summation of n_i over i will give the total number of data instances, N, in the database, and therefore the rank of $\mathbf{S_W}$ can be at most N-C. On the other hand, if $N - C < d$, $\mathbf{S_W}$ will be singular, where d is the dimensionality of the original patterns. ◆

Again, in many signal processing and pattern recognition applications, the number of data instances, N, is much larger than the original dimensionality. However, there are quite a few areas of applications where this does not hold. For example, in the case of ultrasonic signals used in inspection of piping welds, signals of length a few hundred to a few thousand are very common, whereas the total number of signals may not be nearly as many. Similarly, in image processing, the number of pixels in a given image can be, and usually is, much larger than the total number of images available. In summary, although FLD is useful for dimensionality reduction in data sets, it cannot always be used for increasing class separability, particularly if C is small or $N - C < d$.

The next section describes the proposed nonlinear cluster transformation method, for addressing the problem of overlapping clusters. The quantitative measure of effectiveness of the method in achieving this goal is calculated using the same criterion function of the FLD algorithm.

4.4.4 Nonlinear Cluster Transformation (NCT)

Nonlinear cluster transformation is a three-step supervised procedure that attempts to increase the intercluster distances, while preserving intracluster distances and the dimensionality of the pattern vectors. Minor reduction in the intracluster distances is achieved, however, by outlier removal. NCT has no limitations in terms of dimensionality, number of classes, or the total number of patterns in the database.

In the first step, outliers are removed using a distance metric based on the Mahalanobis distance. In the second step, the desired cluster separation is obtained by a simple translation of each cluster along an optimal direction. This step, in essence, generates training data pairs for determining the NCT mapping function of the third step. In this last step, the data generated in step two is used to train a generalized regression neural network (GRNN) to approximate the function mapping between original clusters and the translated clusters. The performance of this algorithm is evaluated by computing the FLD criterion function in the pattern space and feature space. The feature vectors are then input to a classifier of choice. The details of these steps are explained in the following paragraphs.

4.4.4.1 Outlier Removal

The patterns in each class i in the training database are first normalized according to

$$\mathbf{x} = \frac{\mathbf{x}}{\sqrt{\sum_{k=1}^{d} (x^k)^2}} \tag{4.20}$$

where x^k is the k^{th} element of the pattern $\mathbf{x}$, and d is the dimensionality of the patterns.

Outlier removal is performed next, based on the Mahalanobis distances of patterns from the cluster centers. For each cluster i, the Mahalanobis distance of pattern $\mathbf{x}$ in class i is computed as

$$M_D = (\mathbf{x} - \mathbf{m}_i)^T \mathbf{C}_i^{-1} (\mathbf{x} - \mathbf{m}_i) \qquad \mathbf{x} \in \{\mathbf{X}_i\} \tag{4.21}$$

where $\mathbf{C}_i$ is the covariance matrix of the pattern population of the i^{th} class, and $\mathbf{m}_i$ is the mean of this population. M_D can be used as a measure of dispersion within the cluster. Note that the Mahalanobis distance is a better distance criterion then the Euclidean distance. The Euclidean distance simply measures distance from the cluster center, and therefore it cannot be used successfully to detect outliers that are close to the cluster centers. In contrast, Mahalanobis distance does not measure distances to cluster center, but rather distances to the cluster itself as a whole, and therefore it is more suited for outlier detection. Note that outlier removal also provides some intracluster distance reduction.

4.4.4.2 Cluster Translation

This step addresses the problem of closely packed and possibly overlapping clusters. The underlying idea is to translate the clusters appropriately in order to physically separate them. Conceptually, all clusters are thought of as like charged particles, and the magnitude and direction of the translation vector are then derived using the concept of a repulsive force exerted by each cluster i on all other clusters. The procedure is first explained for a two-class problem. The natural extension to the multi-class case is then derived.

Consider a two-class problem with possibly overlapping clusters, whose centers are located at $\mathbf{m}_1$ and $\mathbf{m}_2$. The distance between these two clusters can be increased if the class I patterns are translated along a vector $\mathbf{S}_1 = -(\mathbf{m}_2 - \mathbf{m}_1)$, and class II patterns are translated along

$\mathbf{S_2}=-\mathbf{S_1} = -(\mathbf{m_1}-\mathbf{m_2})$. This idea can be extended to multi-class problems of arbitrary dimensionality, where patterns of class C_i can be translated along $\mathbf{S_i}$, where the optimal direction $\mathbf{S_i}$ can be computed as

$$\mathbf{S}_i = -\sum_{j \neq i}^{C} (\mathbf{m}_j - \mathbf{m}_i) \tag{4.22}$$

and where $\mathbf{m}_i$ and $\mathbf{m}_j$ are the cluster centers of cluster i and cluster j, respectively, and C is the number of clusters.

The resultant translation vector for cluster i is $\mathbf{S}_i = -\mathbf{M}_i$, where

$$\mathbf{M}_i = \left(\sum_{j \neq 1} (\mathbf{m}_j - \mathbf{m}_i) \right) \tag{4.23}$$

and $\mathbf{m}_i$ is the cluster center of the i^{th} cluster.

All patterns in cluster i are moved along the direction of $-\mathbf{M}_i$, and the translated patterns can be obtained by

$$\mathbf{x}_{Si} = \mathbf{x}_i + \left(-\mathbf{M}_i / \|\mathbf{M}_i\|\right) \cdot dist_i \tag{4.24}$$

where $\mathbf{x}_i$ is a pattern from cluster i, $dist_i = 1/|\mathbf{m} - \mathbf{m}_i|$ is a normalizing constant that controls the amount of translation, and $\mathbf{x}_{Si}$ is the new location of the pattern $\mathbf{x}_i$. This intuitive approach is shown in Figure 4.19.

It is straightforward to show mathematically that these translation directions maximize intercluster distances.

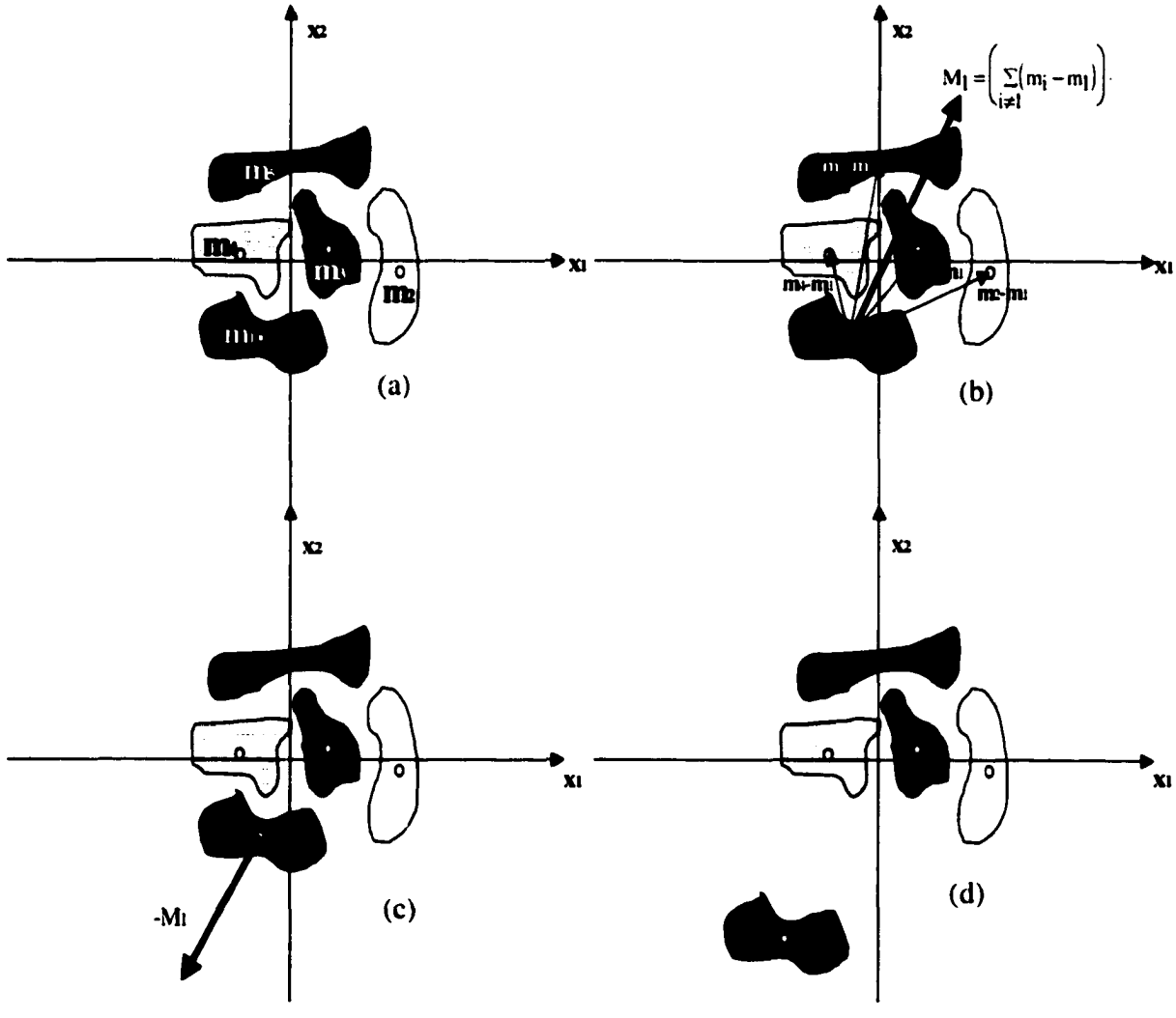


Figure 4.19 An intuitive approach for increasing pattern separability

Theorem 3. *For a C class problem the sum of all intercluster distances defined as in Equation 4.25 is maximized if all clusters are translated along* $\mathbf{S}_i = -\mathbf{M}_i$, $i=1,..,C$, *the opposite of the resultant of the vectors that pair wise connect the cluster centers.*

Proof: For a C class problem, we first define the overall intercluster distance D as the sum of all intercluster distances as shown in Equation 4.25.

$$D = \sum_{i,j=1}^{C} D_{ij} = \sum_{i,j=1}^{C} (\mathbf{m}_i - \mathbf{m}_j)^T (\mathbf{m}_i - \mathbf{m}_j) \tag{4.25}$$

After translation along $\mathbf{M}_i$ according to Equation 4.23, the new overall intercluster distance is

$$D^{new} = \sum_{j=1}^{C} (\mathbf{m}_i + \mathbf{M}_i - \mathbf{m}_j)^T (\mathbf{m}_i + \mathbf{M}_i - \mathbf{m}_j) + \sum_{\substack{j,k=1 \\ j,k \neq i}}^{C} (\mathbf{m}_k - \mathbf{m}_j)^T (\mathbf{m}_{ki} - \mathbf{m}_j) \tag{4.26}$$

The vector $\mathbf{M}_i$ that *extremizes* D^{new} is obtained by setting the derivative of D^{new} with respect to $\mathbf{M}_i$ to zero, and solving for $\mathbf{M}_i$:

$$\begin{aligned} &\frac{\partial D^{new}}{\partial \mathbf{M}_i} = \sum_{j=1}^{C} 2(\mathbf{m}_i + \mathbf{M}_i - \mathbf{m}_j) = 0 \\ &\Rightarrow \mathbf{M}_i = \frac{1}{C} \sum_{j=1}^{C} (\mathbf{m}_j - \mathbf{m}_i) \end{aligned} \tag{4.27}$$

since

$$\frac{\partial^2 D}{\partial (\mathbf{M}_i)^2} = 2C > 0 \tag{4.28}$$

this value of $\mathbf{M}_i$ minimizes the overall intercluster distance. We therefore deduce that the optimal direction of translation, $\mathbf{S}_i$, to *maximize* the new overall intercluster distance must be the opposite of $\mathbf{M}_i$, that is,

$$\mathbf{S}_i = -\mathbf{M}_i = -\frac{1}{C} \sum_{j=1}^{C} (\mathbf{m}_j - \mathbf{m}_i) \tag{4.29}$$

Note that $\mathbf{S}_i$ points in the opposite direction of the resultant vector combining the center of cluster i to the centers of all other clusters, that is, it points away from all other clusters. ◆

The cluster transformation described here can also be expressed in a matrix form. Let $i=1,2,...,C$, where C is the number of classes, $n=1,2,...,N_i$ where N_i is the number of patterns in class i, X_n^i be the n^{th} pattern of the i^{th} class, and Y_n^i be the corresponding pattern after translation. Then,

$$\begin{pmatrix} Y_1^i \\ Y_2^i \\ . \\ . \\ Y_{Ni}^i \end{pmatrix}_{[Ni\times 1]} = -dist_i \cdot \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & & \\ & \vdots & \\ & & \vdots \\ 1 & \cdots & 1 \end{pmatrix}_{[Ni\times C]} \cdot \begin{pmatrix} \mathbf{m}_1 - \mathbf{m}_i \\ \mathbf{m}_2 - \mathbf{m}_i \\ . \\ . \\ \mathbf{m}_C - \mathbf{m}_i \end{pmatrix}_{[C\times 1]} + \begin{pmatrix} X_1^i \\ X_1^i \\ . \\ . \\ X_{Ni}^i \end{pmatrix}_{[Ni\times 1]} \tag{4.30}$$

This equation is implemented on the training data sets to generate a second dataset that is used to train a GRNN to learn the overall transformation function.

4.4.4.3 Function Mapping

In order to translate each cluster away from each other, the cluster information is required, which is not available for test patterns. We therefore need to *learn* how to translate patterns without knowing the class information. This problem can be thought of as a function approximation problem, where the function to be approximated is a function that maps d dimensional original patterns to their new locations. A generalized regression neural network (GRNN) was used to accomplish this function approximation. GRNN, developed and shown to be a universal approximator by Specht [106], can be though of as a special case of radial basis function neural network (RBFNN). GRNNs do not require iterative training, and they can approximate any arbitrary multidimensional function defined between a set of input and

output vectors. Therefore, they have been used with significant success in multidimensional function approximation. GRNN is based on the theory of nonlinear regression analysis, commonly used as a statistical function estimation scheme. As shown below, GRNN is very similar to RBFNN, the only exception being the different procedure used to assign output layer weights. In fact, GRNN and RBFNN have identical architectures, as shown in Figure 4.20.

The method of nonlinear regression analysis estimates the expected value of y as [107]

$$E[y|\mathbf{x}] = \frac{\int_{-\infty}^{\infty} y\, f(\mathbf{x}, y)dy}{\int_{-\infty}^{\infty} f(\mathbf{x}, y)dy} \tag{4.31}$$

where y is the output of the estimator, $\mathbf{x}$ is the input vector for which the corresponding output is to be estimated and $f(\mathbf{x},y)$ is the joint probability density function of $\mathbf{x}$ and y. Specht showed that Equation 4.31 can be optimally approximated as

$$y_j = \frac{\sum_{i=1}^{N} R_i w_{ij}}{\sum_{i=1}^{N} R_i} \tag{4.32}$$

where

$$R_i = e^{-\frac{\|\mathbf{x}-\mathbf{u}\|}{2\sigma^2}} \tag{4.33}$$

is output of the i^{th} receptive field (hidden neuron), w_{ij} is the weight that connects the i^{th} hidden neuron to the j^{th} output neuron, σ is a spread constant that controls the ranges of the receptive regions, and finally $\mathbf{u}$ are the training vectors and they are the centers of the receptive fields [107].

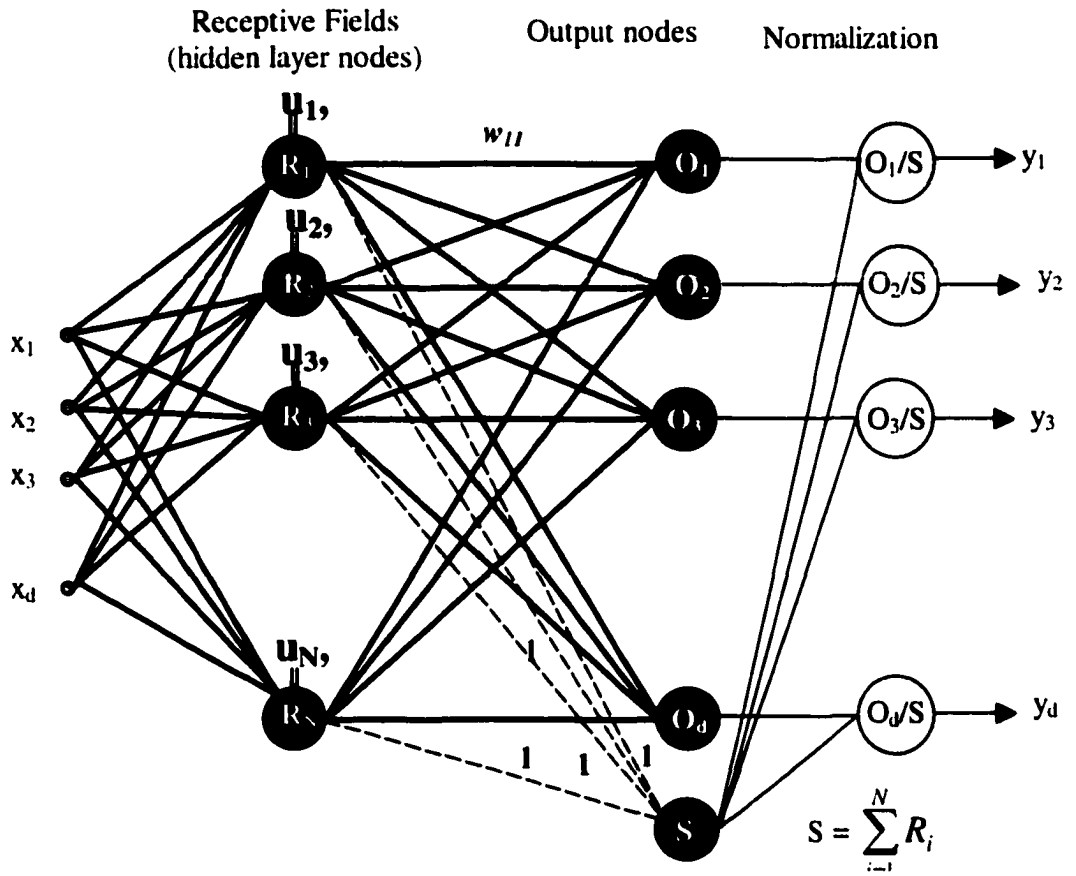


Figure 4.20 GRNN architecture

As mentioned above the only difference between RBFNN and GRNN is the way the weights are determined. In GRNN, the weights w_{ij} are simply assigned as the target outputs of the network. For example, weights connecting the first hidden node to output nodes, w_{1j}, are determined from the values of the target output corresponding to the first training vector. Note that the number of hidden layer nodes is therefore equal to the number of training vectors that are available. GRNN is essentially a scheme for estimating the joint probability density function of input and output from a training dataset. It may be argued that GRNN simply memorizes the training data vectors; however, by a suitable selection of the spread constant,

σ; it is able to generalize arbitrarily complex functions [107]. It is also interesting to note that there is a very close relationship between GRNN and probabilistic neural networks (PNN), the former having an additional normalization at the output layer [107,108].

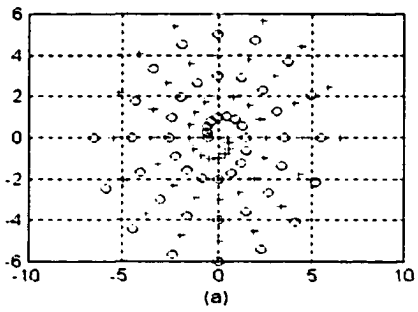
4.4.5 Experimental Results

NCT has been tested on various databases, three of which are discussed in this section. The first database is the double spiral database, which consists of two interleaved spirals, and the second database is an artificially generated two-dimensional multiclass database. The third database is the gas sensing database described previously.

4.4.5.1 Double Spiral (DS) Database

The DS database is a popular database used extensively for evaluating neural network architectures. This database consists of two distinct spirals in the x-y plane, as shown in Figure 4.21. The advantage of this database is that it has a two-dimensional feature space, thus allowing easy visualization of patterns. The task of distinguishing these two spirals is known to be a difficult task for back-propagation networks and their relatives.

The database was divided into a training dataset, T_{DS}, and an evaluation dataset, E_{DS}. The separation was obtained by putting every other instance into T_{DS} and the remaining instances into E_{DS}. The original database had 194 instances, and they were equally divided into training and evaluation datasets of 97 instances each. Figure 4.21(a) illustrates the T_{DS} database, and Figure 4.21(b) shows the result of cluster translation on T_{DS}. These patterns were then used to train a GRNN, which was evaluated on the E_{DS}. Figure 4.22(a) and (b) illustrate the E_{DS} dataset before and after NCT operation. Note that the two spirals are now linearly separable and technically, they do not require a network to distinguish them from each other.



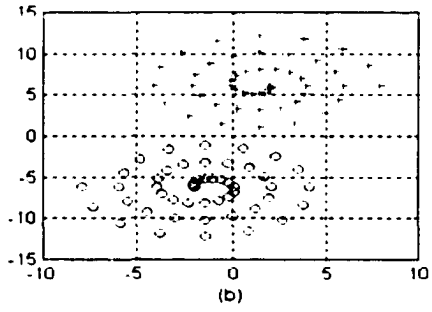
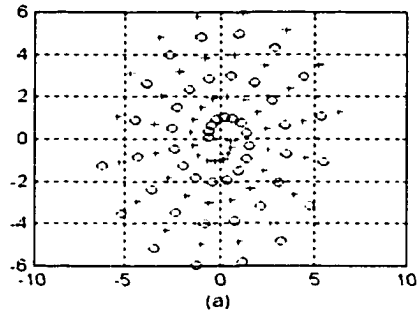


Figure 4.21 (a) T_{DS} dataset
(b) T_{DS} after translation



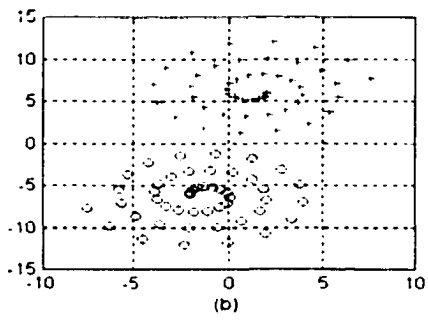


Figure 4.22 (a) E_{DS} dataset
(b) E_{DS} after NCT

4.4.5.2 Synthetic Data

A synthetic dataset with three overlapping classes in the 2-D feature space was generated for a more challenging test. Similar to the double spiral database, this database was divided into training (T_{SYNT}) and evaluation (E_{SYNT}) datasets. Figure 4.23 shows the training data with the corresponding translated targets computed using the equations given above. Two datasets shown in Figure 4.23 were used to train a GRNN to learn this nonlinear mapping. This example shows the effect of the spread constant, σ, in the GRNN formulation. The generalization capabilities of GRNN are shown in Figures 4.24 and 4.25 for various values of σ. As σ decreases, sharper Gaussians are used for approximations. This improves the accuracy and generalization of the GRNN only up to certain values of σ, since further decreasing σ results in the network memorizing the training patterns with no generalization property.

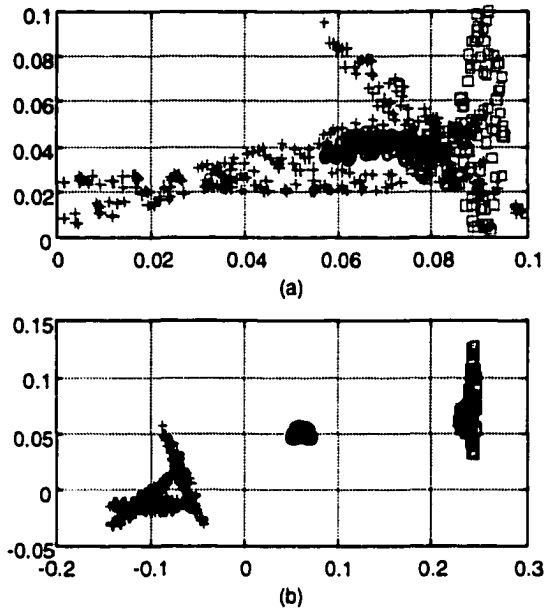


Figure 4.23 (a) T_{SYNT} dataset
(b) T_{SYNT} after translation (target for GRNN)

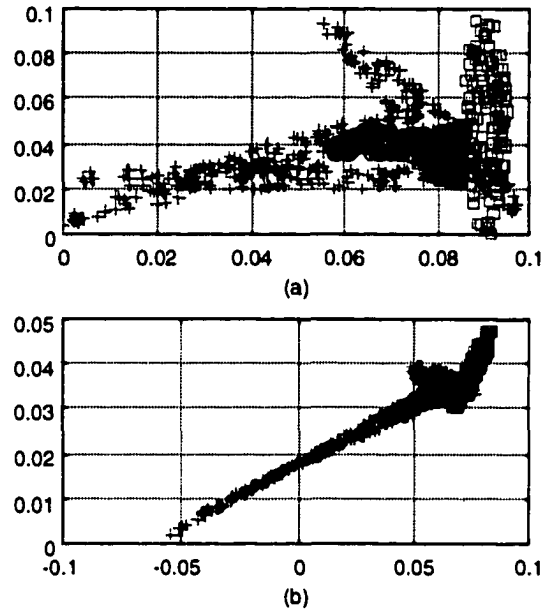
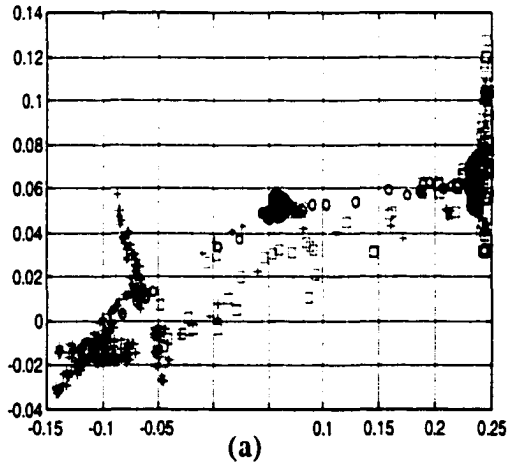


Figure 4.24 (a) E_{SYNT} dataset
(b) E_{SYNT} after NCT, for σ=0.05.



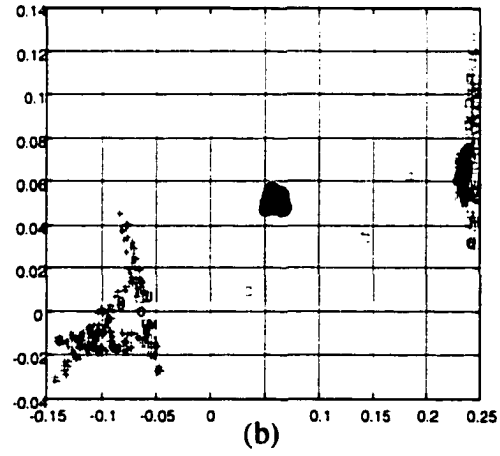


Figure 4.25 (a) E_{SYNT} after NCT, for σ=0.01 (b) E_{SYNT} after NCT, for σ=0.0001

4.4.5.3 VOC Mixture Database

NCT was finally tested on the dominant VOC identification problem. Recall that for the identification of the dominant VOC problem, there were five classes, namely, OC, ET, XL, TL and TCE. For the purposes of visualizing the data, and the effect of NCT, only three attributes were used in the following figures. The computations, however, used all six attributes. As before, the database was partitioned into two parts, T_{VOC} for training and E_{VOC} for evaluation. Each partition had 192 instances. Figure 4.26 illustrates T_{VOC} with its first three attributes, and the result of cluster translation on T_{VOC}. The training database T_{VOC} and its translated target vectors were used to train the GRNN to learn the NCT for this database. GRNN was then evaluated on E_{VOC}. Figure 4.27(a) illustrates the evaluation database E_{VOC}, and Figure 4.27(b) shows the output of the GRNN for this dataset. As we can see from Figure 4.27(b), the transformed patterns do not look quite as well separated (at least in 3-D) as

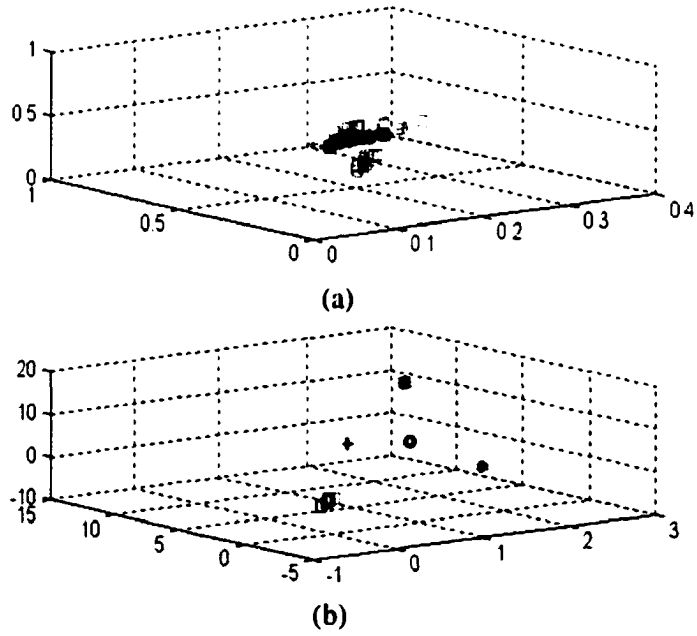


Figure 4.26 T_{VOC} (a) before and (b) after transformation

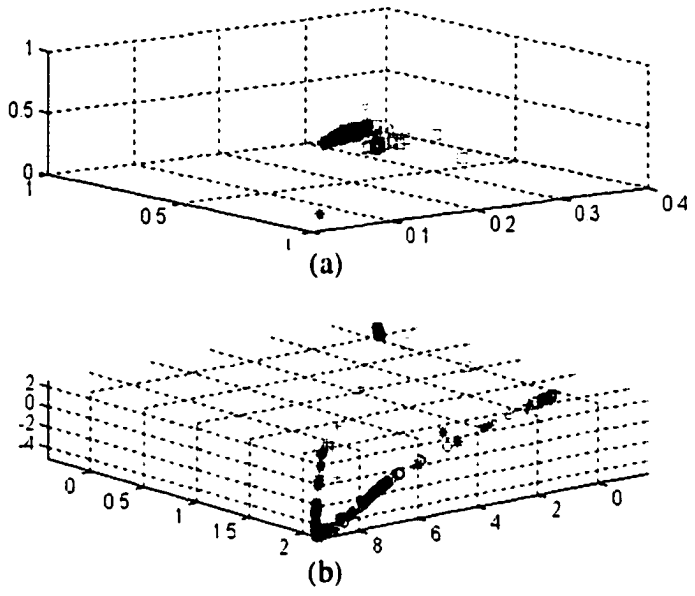


Figure 4.27 E_{VOC} (a) before and (b) after NCT

the training data in Figure 4.26(b), due to the inherent difficulty of this dataset. However, the five clusters in Figure 4.27(b) are better separated (even to the human eye) after the NCT preprocessing, compared to unprocessed patterns of Figure 4.27(a).

Very satisfactory results were obtained when this procedure was used as a preprocessing step for a subsequent classifier, such as a neural network. A three-layer MLP easily converged to a small error minimum when trained with the NCT processed patterns. Repeating over twenty trials with various spread parameters, a correct classification performance of 80%-95% were achieved over the entire E_{VOC} dataset that the MLP had not seen before.

Recall from our previous discussion that neither a similar architecture, nor a larger architecture of MLP or RBF was able to converge, prior to preprocessing of this database. FLD was also tried on this database, which was unable to project the data on to a 4-D space where

the patterns were more separable. It is interesting to note that the J(W) criterion function was evaluated before and after NCT for this E_{VOC}, which showed an increase of seven orders of magnitude after NCT. This demonstrated the effect of NCT on the evaluation database in increasing the intercluster distances.

4.4.6 Conclusions and Future Work for NCT Analysis

The main purpose of NCT is to increase the intercluster distances while keeping intracluster distances constant. Preprocessing with NCT allowed improved performances of subsequent classification algorithms, and in fact, it made training possible in the first place, for the VOC database.

The following comments can be made for the advantages of NCT. Unlike FLD, NCT has no limitation on the number of classes, dimensionality, or the number of instances. NCT can be applied to virtually any database of arbitrary dimensionality with arbitrary number of classes. Training for NCT is a single step procedure which does not require an iterative learning; therefore it is very fast. Although not guaranteed, NCT may provide minor dimensionality reduction for certain databases through outlier removal. On the other hand, NCT has its own limitations. NCT will not work for a database that has two clusters with identical means. However, the distance between them can be arbitrarily small, as was the case in the double spiral database. Furthermore, the speed of the NCT comes at an expense of computational space complexity. NCT require considerably larger memory than iterative learning techniques, particularly for databases with a large number of data instances.

Future work for this algorithm includes an improved translation criterion that not only considers the distance between clusters, but also their variances. Such a scheme has the po-

tential to improve the overall performance of this scheme in two ways. First, it would allow us to determine the optimum amount of translation for patterns of different classes, and second it would allow us to move patterns of the same class towards each other, hence providing reduction in the intracluster distances as well as increase in the intercluster distances.

4.5 Conclusions on Enhancing Pattern Separability

Three approaches for identifying VOCs in mixtures, in particular the dominant VOC, were presented in this chapter. The first approach made use of a fuzzy inference system to identify the dominant VOC and a neural network to identify the secondary VOC. The most important characteristic of this FIS was its selection of membership functions that made use of dynamic ranges of individual sensor responses, and the consequent design of its rule base. About 89% of all mixtures were classified with their correct dominant VOCs by FNOSE, which was followed by secondary VOC neural networks, one for each dominant class. The overall system had a classification performance of 83%.

One main advantage of fuzzy inference systems over neural networks is that fuzzy systems are very intuitive and straightforward to design and very simple to interpret, in contrast to neural networks which are black boxes to the end user. This is due to their massively parallel and complicated inner structure. However, this massively parallel and complicated structure can be a very powerful classifier when trained with appropriately preprocessed patterns.

The second approach, proposed as an appropriate preprocessing scheme, was inspired by the scheme used in the selection of membership functions for FNOSE, and the histogram equalization scheme used in image processing. The outputs of all sensors were individually

transformed in an attempt to increase their dynamic range. The mapping functions were selected based on the sorted histograms of sensor outputs, and these functions were then implemented by RBF neural networks. When *feature range stretching (FRS)* scheme was applied to mixture signals, both the dominant VOC network and the secondary VOC networks correctly classified 96% of the patterns. The classification performance of the cascaded system was 92%. The FRS preprocessing increases the intercluster distances of the signals to allow better separability of the data. However, it should be noted that this scheme also increases the intracluster distances, which is undesirable. Another disadvantage of the FRS technique is that it requires a stretching function to be manually formulated for each feature in the pattern. This can be a tedious process for databases of high dimensionality.

The third approach, preprocessing with nonlinear cluster translation, was therefore developed to address the issue of increasing the intercluster distances without changing the intracluster distances. Based on translating the clusters away from each other and learning this mapping using a GRNN, this approach offers performance comparable or superior to the previous methods, but without the implementation problems. Note that the mapping of each feature is automated in this procedure, and hence it does not suffer from similar implementation issues that FRS does. The only parameter that needs to be determined by the user is the spread constant for the GRNN.

It should be emphasized that despite its relatively poorer performance compared to FRS and NCT preprocessing schemes, the fuzzy inference system approach has proven viable in identification of the dominant VOCs, and therefore warrants detailed investigation for performance improvement. It should be noted that FIS achieved its classification performance of

89% in the classification of the dominant VOCs without requiring any preprocessing, a noteworthy achievement unattained by any neural network tested to date.

CHAPTER 5

OPTIMUM FEATURE SELECTION

5.1 Introduction and Motivation: Knowing What Doesn't Matter

As discussed in Chapter 3, piezoelectric chemical sensors, such as surface acoustic wave (SAW) devices and quartz crystal microbalances (QCMs) have been widely used for the detection and identification of volatile organic compounds (VOCs), where an array of polymer coated sensors is generally used [39, 68, 69, 70, 76, 78, 109]. The change in the resonant frequency of each sensor as a function of VOC concentration constitutes the response pattern. Over the past fifteen years, a significant amount of work has been done on developing pattern recognition algorithms, using principal component analysis, neural networks and fuzzy inference systems, for various gas sensing problems [3, 18, 19, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36]. However, these methods can only be successful, if the features (polymer coated sensor responses) used to identify the VOCs allow efficient separation of patterns in the feature space. The challenge is then to identify a subset of polymer coatings such that a classification algorithm provides optimum classification performance. Selection of coatings is usually based on various chemical properties (e.g., solubility parameters [44, 45, 71]) of the VOCs and the compatibility of each with a range of compositionally different polymer coatings.

Since there may be a large number of polymers suitable for the identification of a VOC, the selection of the smallest set giving the best performance is an ill-defined problem. This is because testing every possible combination is usually not feasible. Many researchers have

observed that using as many sensors as possible does not necessarily improve the performance of a classification system. In fact, Park *et al.* [110,111] through a careful analysis of the required number of sensors versus the number of analytes and Osbourn *et al.* [112] through steadily increasing the sensor size, have shown that the performance of classifiers for VOC identification typically degrade, as the number of sensors increase beyond a certain number. Therefore, an efficient algorithm for optimum selection of sensors is of paramount importance.

For small pools of potential coatings, an exhaustive search may be manageable. For example, Zellers *et al.* [22] successfully conducted an exhaustive search of a ten-polymer dataset that used extended disjoint principal components regression analysis to evaluate classification performance. Using this strategy, four polymers were identified as requisite array elements for optimum identification of six VOCs [22, 110, 111]. The use of four polymers out of ten amounts to 210 possible combinations, which is manageable for an exhaustive search. However, as the number of possible coatings increases, using exhaustive search becomes computationally prohibitive. An addition of two more coatings, for instance, to the pool requires evaluating 495 possible four-coating combinations, and a more practical problem of choosing six coatings out of twenty coatings requires testing 38760 different combinations of coatings.

In an effort to reduce the number of candidate coatings from a larger pool of potentially useful coatings, various feature extraction and dimensionality reduction algorithms have been developed. Principal component analysis (PCA), as described earlier in Chapter 4, has been one of the most popular of such techniques. Carey *et al.* [113] used PCA to reduce the feature vector obtained from 27 sensors to less than 8 for an identification problem consisting of 14

VOCs. Avila *et al.* [114] introduced correspondence analysis as an alternative to PCA and showed that it had computational advantages as well as performance improvement over PCA on the same dataset used by Carey *et al.*

In PCA the strategy is to find a set of n orthogonal vectors along which the m dimensional data has the largest variance such that $n<m$. PCA is, therefore, a *dimensionality reduction* procedure, rather than a *feature selection* procedure. The principal components are computed as the projection of the data onto a set of orthogonal vectors that are the eigenvectors of the covariance matrix of the data. The covariance matrix may, and frequently does, contain significant information obtained from each sensor. Consequently, PCA does not reduce the number of sensors, nor does it identify the optimum set of coatings. Recently, Osbourn *et al.* and Ricco *et al.* [98, 99] pointed out the limitations of PCA for feature selection as well as for identification and introduced Visual Empirical Region of Influence Pattern Recognition (VERI-PR) for identification of VOCs, which was discussed in Section 4.1 of Chapter 4. The authors state that VERI does not suffer from various shortcomings present in other techniques. For example, VERI does not require or assume any specific probability distributions to be known, and it does not require a large number of parameters to be adjusted by the user. Furthermore, VERI is a versatile algorithm not only capable of pattern recognition, but also of optimum feature selection. The optimal feature selection capabilities of VERI on a VOC identification problem have been reported to be very promising [100]. However, the feature selection module is based on an exhaustive search, called *leave-one-out*, and therefore the authors recommend its use for pools of less than 20 coatings. Despite the drawbacks of PCA and development of new algorithms, PCA and related techniques are still being used in feature extraction for gas sensing applications [115, 116, 117].

Selection of optimum coatings for gas sensing is an example of the more general problem of choosing an optimum subset of features, which is commonly encountered in pattern analysis, machine learning and artificial intelligence [94, 118]. It is interesting to note that in many pattern recognition and classification applications incomplete or inadequate data sets are usually blamed for poor performance. In the context of gas sensing, an inadequate data set often refers to having an insufficient number of sensors because it is usually believed that increasing the number of sensors would offer better separability. In fact, the assumption that increasing the number of features also improves classification performance is a misconception. This situation would only be true if the number of responses were increased drastically with the number of features. It may in fact be rather surprising that poor performance might sometimes be due to too much *irrelevant* information. Many studies have shown that most classification algorithms perform best when the feature space includes only the most relevant information that is required for identification [119, 120, 121].

While having relevant features is a key to the successful performance of any classification algorithm, the definition of a relevant feature has been extensively debated. Some studies suggest algorithms that are preprocessing in nature. These preprocessing algorithms *filter* the data, and eliminate irrelevant features. Statistical measures, such as properties of the probability distribution function of the data, are employed for designing the appropriate filter, and therefore, these algorithms are referred to as filter approaches [122, 123, 124].

Etemad and Chellappa's work [125, 126] on feature selection is one of the recent studies that fall into this category. They used wavelet packets [127] to extract useful features. Multi resolution wavelet analysis has been used for data compression (dimensionality reduction) with significant success due to its capability for extracting localized spectral information. In

this work, the authors have used wavelet packets along with a variety of quantitative measures of separability criteria to obtain a minimally optimum set of features. Such criteria include Bayes risk, and within and between-class scatter matrices (see Equation 4.16). At each decomposition level, they have computed these measures and discarded the features that did not contribute significantly to these measures.

The main disadvantage of filtering schemes is that they completely ignore the learning algorithm to be used with the selected features, and selection is solely based on data. Some researchers suggest that set of relevant features for any data should be dependent on the classification algorithm [118, 120, 128, 129]. For example, a good set of features for a neural network may not be as effective for decision trees. Such schemes in which the feature selection algorithm is based on or *wrapped around* the classification algorithm are referred to as wrapper approaches [129]. Most algorithms in this class employ a heuristic search algorithm to find the minimum set of optimum features; therefore, they suffer from prohibitively large computational time and space complexity problems due to the additional overhead of evaluating the classification performance on each feature subset. Various remedies have been proposed to speed up the search criterion. For example, using a random selection of attributes resembling the simulated annealing algorithms [130], or genetic algorithms [128] instead of an organized search have been investigated. Other researchers have tried training a neural network and then pruning the node corresponding to non-contributing inputs [131], or using statistical measures to start the search from a point closer to the final target (see Section 5.5 in this chapter).

Due to the limited number of possible coatings typically used in the gas sensing area, the computational complexity of wrapper approaches does not constitute a major drawback. We

have therefore analyzed two techniques based on the wrapper approach, and we report the performances of these two artificial intelligence (AI) approaches for selecting the optimum set of coatings for VOC identification. The first approach is based on Quinlan's ID3 (Iterative Dichotomizer 3) algorithm [119], a decision tree algorithm that integrates classification and feature selection. The second approach is a modified version of the wrapper model of Kohavi *et al.* [129], which uses a hill-climb search algorithm to search the feature space for an optimum set of features. The original wrapper model combines the hill climb search with ID3. In this work, the hill-climb search has been integrated with a multilayer perceptron (MLP) neural network. To accelerate the convergence of the hill-climb search, basic statistical indicators have also been incorporated to find a different starting point that would give the search a jump-start. This scheme significantly reduced the computational complexity of the search.

We emphasize that our goal was to develop a systematic and efficient *procedure* for determining the optimum coatings. The analytes and coatings used in this study were selected from those that have been reported extensively in the literature.

5.2 Experimental Setup and Data Handling

The experimental setup used for this study was identical to that described in Chapter 3. Figure 3.4 illustrates this setup. However, an array of twelve crystals was used to detect and identify twelve VOCs, as opposed to an array of six crystals used for the mixture VOC experiment. The polymers used as coatings were as follows:

APZ: Apiezon, **PIB**: Polyisobutelene, **DEGA**: Poly(diethyleneglycoladipate),

SG: Solgel, **OV275:** Poly(ethyleneglycoladipate), **PDS**: Polydimethylsiloxane

PDPP: Poly(diphenoxylphosphorazene), **PCP**: Polychloroprene,

PDS-CO: Polydimethylsiloxane-co-methyl(3-hydroxypropyl)siloxane-graft-poly-(ethylene glycol)3-aminopropylether,

PDS-OH: Polydimethylsiloxane, hydroxy terminated, **PSB**: polystyrene beads,

GRAP: graphite.

The twelve VOCs used were similar to those described in Chapter 3, namely, acetone (AC), methylethylketone (MEK), ethanol (ET), methanol (ME), dicholoroethane (DCA), acetonitrile (ACN), trichloroethane (TCA), trichloroethylene (TCE), hexane (HX), octane (OC), toluene (TL) and xylene (XL). These VOCs were exposed to the sensor array at seven different concentration levels, namely at 70, 140, 210, 250, 300, 350 and 700 parts per million (ppm), giving 84 responses, constituting the experimental database used in this study. These responses were considered as signature patterns of their respective VOCs, and thus they were used as representatives of the corresponding VOCs.

The responses of the sensors to the given VOCs were eminently linear with the concentration of these VOCs in the given concentration range. Therefore, the available data with responses to twelve VOCs at seven concentrations were interpolated and extrapolated to include fifteen concentrations using regression analysis. Linear regression coefficients with $r^2>0.998$ were obtained for every sensor. The augmented data set allowed us to obtain estimated frequency responses of the sensors at the following concentrations 70, 100, 150, 200. 250, 300, 350, 400, 450, 500, 600, 700, 800, 900, 1000, and 1500 ppm, giving 12*15=180 data instances. This data, however, was not used for any of the training algorithms, but it was simply used for testing neural network performances. The rationale for generating such a data set to test the performance of the neural networks on an expanded data set was to measure its performance under noisy conditions. Since this data set is artificially generated, the estimated

frequency responses can be considered as approximations to what the actual responses should be. Therefore, such a database can be viewed as a candidate for a noisy database.

It has also been realized that the change in frequency is linearly dependent on the thickness of the coatings: the thicker the coating, the higher the frequency response. However, no attempt has been made to date to normalize the data with respect to the coating thickness in order to test the generalization capabilities of the neural network classifiers.

In the following sections, the frequency response patterns of the QCM array to various VOCs are referred to as *feature vectors*. The response of each individual sensor coated with a different polymer constitutes the individual *features*. We therefore use the terms *feature* and *sensor response* interchangeably. Furthermore, we will use the abbreviations of the coating names to refer to individual features where necessary.

5.3 Method I: ID3 / C4.5 / C5.0 Family of Decision Trees

5.3.1 Generating Decision Trees

Decision trees are compact forms of displaying a list of IF-THEN rules in a hierarchically ordered fashion of a tree structure. These rules are used to make classification decision about the input pattern. Decision trees are one of the most commonly used machine learning algorithms for classification applications, ID3 being one of the most popular among all [118]. In fact, ID3 (and its improved descendants, C4.5, C5.0) has become the basis of a general class of decision tree algorithms, referred to as ID3 based algorithms. ID3 is mainly a classification algorithm that classifies test data (validation data) by constructing a decision tree from training data [119]. The algorithm determines the features necessary for correct classification of training data. The decision tree starts with what ID3 identifies as the most important fea-

ture based on the information content of each feature. Given a training data set, T, the probability that a certain pattern belongs to a given class, C_i, is given as

$$P = \frac{\text{frequency}\ (C_i, T)}{|T|} \tag{5.1}$$

where frequency (C_i, T) is the number of patterns in the training set T that belong to class C_i, and $|T|$ is the total number of patterns in the training data set. The information provided by Equation 5.1 is defined as

$$I = -\log_2\left(\frac{freq(C_i, T)}{|T|}\right) \tag{5.2}$$

and measured in units of *bits*. Note that, by definition of probability, P must be between and zero and 1, the logarithm of which is always negative. The minus sign in Equation 5.2 assures that information is defined as a positive quantity.

The average amount of information, info (T), needed to identify the class of any pattern, in training set T is then defined as the sum over all classes weighted by their frequency of occurrence, or by their probability:

$$\text{info}(T) = -\sum_{i=1}^{N}\left(\frac{freq(C_i, T)}{|T|}\right) \times \log_2\left(\frac{freq(C_i, T)}{|T|}\right) \text{bits} \tag{5.3}$$

The information needed to identify the class of any pattern after the training data set has been partitioned into K subsets based on the value of some feature X is given by

$$\text{info}_X(T) = \sum_{i=1}^{K}\frac{|T_i|}{|T|} \times \text{info}(T_i) \tag{5.4}$$

where $|T_i|$ is the number of patterns in the partition i. Equation 5.3 is generally referred to as the entropy before partitioning, and Equation 5.4 as the entropy after partitioning of the train-

ing set T. The original ID3 algorithm uses *gain* as the criterion to determine the additional information obtained by partitioning T using the feature X. Thus

$$\text{gain}(X) = \text{info}(T) - \text{info}_x(T) \tag{5.5}$$

ID3 first selects the feature that has the largest gain and places that feature at the root of the tree. This feature is then removed from the feature set, and the feature that has the largest gain among the rest becomes the second important feature and so forth. This criterion, however, has a very strong bias in favor of features that have many outcomes, that is, features whose values partition the data set into most number of classes. Although this may seem a logical bias, it will cause the classifier to produce poor performance if an irrelevant feature uniquely identifies all classes. An example of such a database is a medical database in which a diagnosis (classification) for each patient (pattern) is made based on the results of a certain number of tests (features). Such databases often include a patient identification number, and this number uniquely matches the diagnosis for that patient. Obviously, the identification number is not actually useful for the diagnosis, but because it uniquely identifies each patient's diagnosis, the gain criterion of ID3 will choose this feature as the root, and will terminate the tree.

The second generation of ID3, named C4.5, overcomes this potential problem by defining *split_info* of a feature:

$$\text{split\_info}(X) = \sum_{i=1}^{K} \frac{|T_i|}{|T|} \times \log_2\left(\frac{|T_i|}{|T|}\right) \tag{5.6}$$

where **split_info** defines the potential amount of information that is generated by dividing T into K partitions by the feature X. Then

$$\text{gain\_ratio}(X) = \frac{\text{gain}(X)}{\text{split\_info}(X)} \tag{5.7}$$

defines the proportion of useful information generated by the split of T. If the number of partitions, K, generated by the feature X is excessively large, split_info(X) will also be large, making the gain_ratio(X) small. C4.5 uses gain_ratio as the criterion for choosing the features.

Details of this procedure, as well as examples, can be found in Quinlan's reference book on C4.5 [132]. C4.5, and more recently C5.0, the newest version of the ID3 family of decision trees, add a number of new features to the algorithm, such as cross-validation and boosting, as described later in this section. A sample decision tree generated by C5.0 for this particular problem of determining optimum coatings is shown in Figure 5.1.

The decision tree given by C5.0 can easily be converted into a set of rules, and a classification can be made from these rules. For example, the first rule generated by the tree in Figure 5.1 can be expressed as " ***IF PIB response is less than 0.19947, AND the response of PSB is less than 0.057534, THEN the VOC is ethanol (ET)***". The number in parenthesis (7.0) refers to the number of patterns that were classified correctly with this rule. In this case, all seven responses (to seven concentrations) of ethanol were successfully classified by this rule. If applicable, a second number following a slash sign is given corresponding to the number of misclassification cases. Also note that these numbers are given in floating point format, implying that they may also be non-integers.

```
PIB <= 0.19947:
:...PSB <= 0.057534: ET (7.0)
:   PSB > 0.057534:
:   :...PDS > 0.058451:
:       :...GRAP <= 0.48869: ACN (7.0)
:       :   GRAP > 0.48869: ME (7.0)
:       PDS <= 0.058451:
:       :...APZ <= 0.063905: AC (7.0)
:           APZ > 0.063905:
:           :...SG <= 0.061274: DCA (5.0)
:               SG > 0.061274: MEK (7.0)
PIB > 0.19947:
:...OV275 <= 0.049858:
    :...PIB > 0.75788: OC (7.0)
    :   PIB <= 0.75788:
    :   :...PDS <= 0.07377: TCE (7.0)
    :       PDS > 0.07377: HX (7.0)
    OV275 > 0.049858:
    :...PSB <= 0.89462: XL (7.0)
        PSB > 0.89462:
        :...DEGA > 0.20139: DCA (2.0)
            DEGA <= 0.20139:
            :...PCP > 0.2028: TL (3.0)
                PCP <= 0.2028:
                :...PDS <= 0.062143: TL (5.0/1.0)
                    PDS > 0.062143: TCA (6.0)
```

Figure 5.1 Sample decision tree generated by C5.0

Non-integer number of cases can appear if a value of a feature is not known (missing data), or if there is overlap among the patterns in the pattern space described by selected features. In such cases the algorithm may split the classification of a pattern into more than one rule, and assigns a fraction of a number as the number of correct classification by each rule.

This decision tree algorithm was tested on the original (experimentally obtained) data set consisting of responses of 12 sensors to 12 VOCs. A number of trees were constructed using various options of C5.0. These options, such as pruning, cross-validation, boosting, etc. were the additions to the original ID3 algorithm as new versions such as C4.5 and C5.0 were developed.

Pruning is a procedure for removing the redundancy from the generated decision tree. Pruning usually results in much simpler trees, using a significantly smaller number of features than the original tree. The features that are used in the final tree are considered as the most important features that by themselves are adequate to classify the entire data set. Cross-validation is used to optimize the generated tree by evaluating the tree on a test data set. This is achieved by partitioning the entire database (training and testing) into M blocks, where each block is internally divided into a training sub-block and a testing sub-block. The testing sub-block is also called the holdout set. During this partitioning, number of patterns and class distributions are made as uniform as possible. M trees are generated from these M blocks of data, and the average error rate over the M holdout sets is considered to be a good predictor of the error rate of the tree built from the entire data. Finally, boosting is also a procedure of generating multiple trees, where the misclassified test signals of the previous tree are moved to the training data set. Please see Chapter 6 for more information on boosting.

5.3.2 Results Using Decision Trees

Among the many trees generated using these various options, none was able to perform to our satisfaction. Although trees were able to reduce the number of features from 12 to around 5-7, the correct classification performance using these features was in the range of 63% to 83%. Previous studies have shown, however, contrary to its intention, this algorithm is most beneficial when the features selected in its decision tree are actually used to train a neural network [133]. That is, this algorithm appears to be a good feature selection algorithm rather than a classification algorithm, although it was originally designed as a classification scheme.

```
PIB <= 0.16172:
:...PDS <= 0.058451:
:   :...APZ <= 0.063905: AC (7.1)
:   :   APZ > 0.063905: MEK (7.3/1.4)
:   PDS > 0.058451:
:   :...PDS <= 0.10796: ACN (7.1)
:       PDS > 0.10796: ME (8.1/3.7)
PIB > 0.16172:
:...OV275 > 0.19642: DCA (12.6)
    OV275 <= 0.19642:
    :...PSB <= 0.89462: XL (8.3/3.1)
        PSB > 0.89462:
        :...OV275 <= 0.049858:
            :...PDS <= 0.07377: TCE (5.2)
            :   PDS > 0.07377: HX (4.7/0.6)
            OV275 > 0.049858:
            :...PDS <= 0.054348: TL (3.7)
                PDS > 0.054348: TCA (10.8/2.7)
```

Figure 5.2 Decision tree generated by C5.0

One of the better trees, constructed using boosting, pruning, and cross validation options, is the five-feature tree shown in Figure 5.2. As seen from Figure 5.2, this tree used features (coatings) APZ, PIB, OV275, PSB, and PDS. These features were used to train a neural network. A multilayer perceptron (MLP) neural network with the 5x25x12 architecture was used. The responses of five sensors constituted the input layer. There were 25 hidden layer nodes and 12 output nodes, each output representing one of the 12 VOCs. The training data were obtained by randomly selecting 30 patterns from the database of 84 patterns. The results are summarized in Table 5.1.

The ***Train*** column indicates the number of signals used in the training data for each VOC, and the ***Perf*** (performance) column indicates the number of correctly classified patterns for each VOC out of seven that were in the original database. With four misclassified signals giving a test data classification performance of (54-4)/54=93%, this neural network performed significantly better than the best decision tree used as a classifier. The same neural

network was then tested on the expanded (synthetic) data set, and the results are shown in Table 5.2. With eight misclassifications, the network that was trained with patterns of the original (experimentally obtained) database, had a correct classification performance of (180-8)/180 = 96 % on the expanded data set.

This classification performance of the neural network, and consequently that of the feature selection capability of the decision tree method, must be evaluated with some skepticism. Recall that the feature subset containing five features was the best of over 40 different feature subsets suggested by C5.0 at various attempts. A number of different parameters were tweaked in order to obtain this feature set. Therefore, this algorithm may not be the most efficient one to use, particularly for novice users who are not very familiar with the decision tree algorithms.

Table 5.1 Results of the neural network trained with the features suggested by C5.0

VOC	Train	Perf.		VOC	Train	Perf.
AC	2	6/7		**TCA**	2	7/7
MEK	1	6/7		**TCE**	3	7/7
ET	4	6/7		**HX**	4	7/7
ME	2	7/7		**OC**	2	7/7
ACN	2	6/7		**TL**	3	7/7
DCA	3	7/7		**XL**	2	7/7

Table 5.2 Results on the expanded data set (15 concentrations)

VOC	Perf.		VOC	Perf.
AC	14/15		TCA	13/15
MEK	14/15		TCE	15/15
ET	10/15		HX	15/15
ME	15/15		OC	14/15
ACN	14/15		TL	13/15
DCA	12/15		XL	13/15

5.4 Method II: Modified Wrapper Approach

The decision tree based approach for the selection of optimum features is a very fast algorithm, since it takes very little time for a decision tree to converge to a solution. However, it is difficult to use, since it requires many parameters to be adjusted as explained later in this section. This makes decision tree based techniques less than appealing to users who are not closely familiar with the algorithm. Furthermore, since a decision tree does not give satisfactory performance as a classifier, a separate classification algorithm, such as a neural network, needs to be used for the actual classification. Recall, however that the features chosen by the decision tree are the optimum features *for decision tree classification* and may not always be optimum for neural network classification. Most importantly, the decision tree does not allow the user to choose the number of features to be selected. Pre-specifying a number K for the number of features, and being able to ask the algorithm to find the best K features that would optimize the performance among all other K feature subsets is a very desirable property. These concerns motivate the search for an alternate method for feature subset selection.

Recently developed wrapper approaches [129, 134] have been successfully used as feature selection algorithms, where the features are selected based on the performance of the subsequent classification algorithm. Thus, the features selected are optimum for a particular classification algorithm. Wrapper approaches also allow us to specify the number of features we would like to select, and there are relatively fewer parameters to choose than in decision trees, making this approach easily accessible to inexperienced users. These benefits, however, come at the cost of computational complexity. In the next subsection, we briefly describe the wrapper approach, followed by the results obtained using this approach.

5.4.1 Strong and Weak Relevance

Kohavi and John [129,134] expanded the meaning of relevance in feature selection by defining *strong relevance* and *weak relevance* as follows: Let X_i be a feature, $S_i = \{X_1,...,X_{i-1}, X_{i+1}, ... X_m\}$ be the set of all features except X_i, and let x_i and s_i be the value assignments to X_i and S_i, respectively. Then, the feature X_i is strongly relevant if, and only if, there exists some x_i, y, and s_i such that for $P(X_i = x_i, S_i = s_i) > 0$

$$P(Y = y \mid X_i = x_i, S_i = s_i) \neq P(Y = y \mid S_i = s_i) \tag{5.8}$$

where Y is a random variable representing class information, and y is a class assignment of the pattern x. A feature X_i is weakly relevant, if it is not strongly relevant, and there exists a subset of features S_i' of S_i for which there exists some x_i, y, and s_i' with nonzero probability, $P(X_i = x_i, S_i' = s_i') > 0$, such that

$$P(Y = y \mid X_i = x_i, S_i' = s_i') \neq P(Y = y \mid S_i' = s_i') \tag{5.9}$$

According to the above definitions, a feature is strongly relevant if removing this feature results in performance degradation of an optimal Bayes classifier. A feature, X, is then

weakly relevant, if it is not strongly relevant, and there exists a subset of features, S', which does not include X, such that the performance of Bayes classifier on S' is worse than the performance on $S' \cup \{X\}$.

Kohavi and John's approach, originally developed to improve the classification accuracy of C4.5, simply searches feature space in an organized manner. The algorithm is based on testing all feature subsets within a limited search space using C4.5 until there is no further improvement in classification performance. The feature subset selection algorithm works together with, or is "wrapped around", its intended classifier.

The best method for finding the optimum feature subset is to search the feature space exhaustively for every possible feature combination. The problem is, however, such a search algorithm can be computationally prohibitive. Because of this problem, only a subset of the feature space must be searched in an organized manner by exploiting any additional information that is available. The search can start, for instance, using all features and progress by removing features that do not contribute notably to the classification performance (backward search). Alternatively, the search can begin with no features and proceed by adding features that contribute the most to classification performance (forward search).

We adopted the forward search approach, where we started the search with zero features. The performance was then evaluated for each feature. Once the feature that gave the best performance was identified, only one feature was added to the search at each iteration. These two-feature subsets were then evaluated by the classifier at each iteration, and the best two-feature subset was determined. A third feature was then added to those two features, and this procedure was continued until adding new features did not improve performance. Kohavi *et*

al. suggested that this method could find all strongly relevant features as well as some weakly relevant features [129].

The method described above is known as the *hill-climb* search algorithm where a subset of feature space is searched until the best performance is found. The problem with this search scheme is that it may get trapped at a local performance maximum and never locate the best feature subset. Frequently, however, even a sub-optimal selection of features can render adequate separation of the data.

Minor modifications were made to the original wrapper approach that would reduce the possibility of sub-optimal search results. We used the classification performance of an MLP neural network as the evaluation function. MLPs are generally capable of solving more difficult classification problems compared to decision trees, due to their massively parallel nonlinear structure, thus improving the performance of a possibly sub-optimal feature set. Further modifications are discussed in Section 5.5.

It should be noted that an organized search of a subset of the feature space is essential, since searching the entire feature space would be computationally prohibitive for datasets with large number of features. For example, searching for the best feature subset from a set of 12 features requires evaluating (that is, training and testing)

$C\left(\frac{12}{1}\right)+C\left(\frac{12}{2}\right)+C\left(\frac{12}{3}\right)+\cdots+C\left(\frac{12}{11}\right)+C\left(\frac{12}{12}\right)=4095$ different networks, where $C\left(\frac{n}{k}\right)$ is the number of possible combinations of choosing k features from a set of n. The maximum number of subsets searched using hill-climb search, on the other hand, is $N(N-1)/2$, where N is the number of features. For 12 features, there would be only 66 subsets to search. Figure 5.3 shows the complete search space for $N=4$, where each node has a binary code

indicating the features that are included and the ones that are not. For instance, the feature subset [0 1 1 0] includes the second and third but not the first and fourth features. Note that each node is connected to nodes that have only one feature added or deleted. Moving from one node to the next is referred to as *expanding*. The hill-climb search algorithm is given in Figure 5.4.

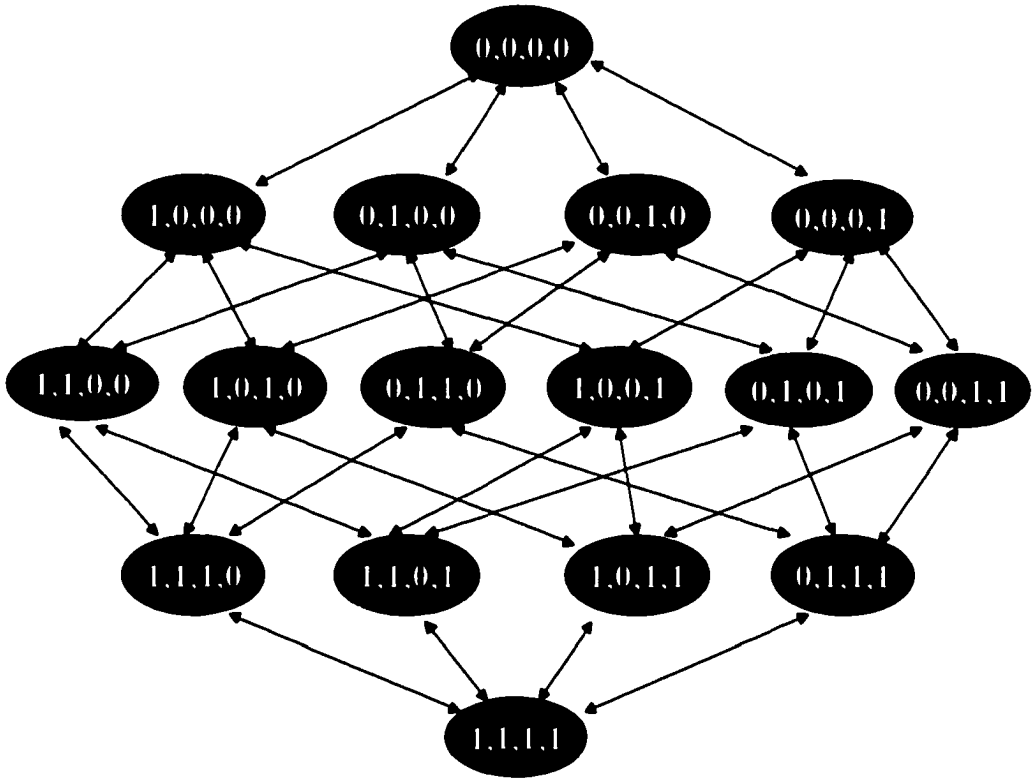


Figure 5.3 Complete feature space for N=4

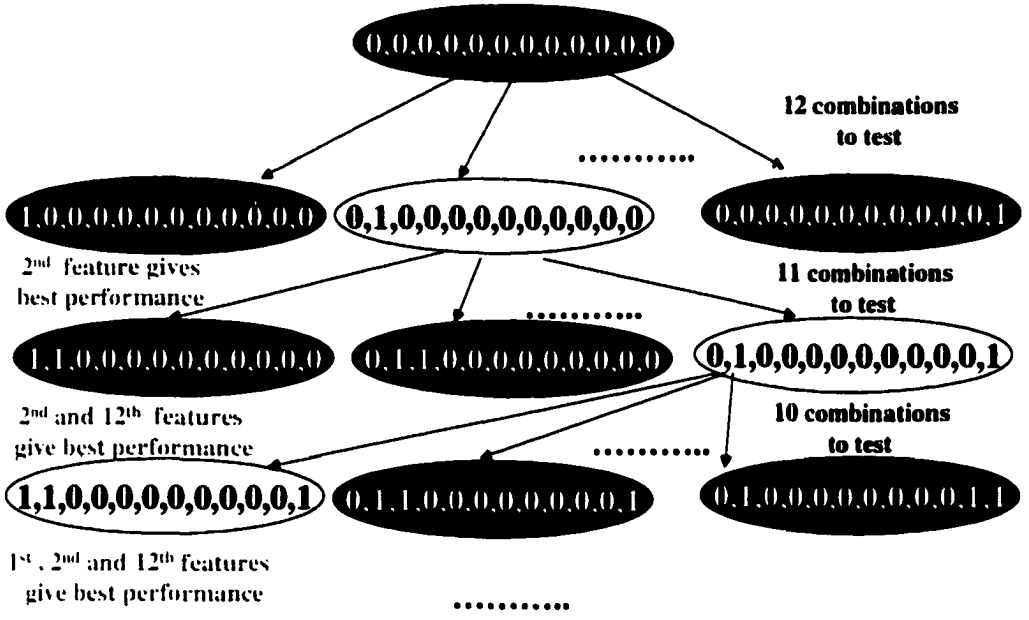
Figure 5.5 illustrates the application of this algorithm to the 12-dimensional feature space. Note that at each stage, the number of possible combinations that need to be evaluated decreases by one. Therefore the total number of feature subsets that must be evaluated is

$$N+(N+1)+(N+2)+\cdots+(N-(N-1))=\frac{N(N-1))}{2}.$$

Algorithm Hill-climb search

1. Let S be the initial feature subset, typically (0,0,...,0,0)
2. Expand S: Find all children of S by adding or removing one feature at a time.
3. Apply the evaluation function, f, to each child, s
4. Let s_{max} be the child with the highest evaluation $f(s)$.
5. If $f(s_{max}) > f(S)$, then $S \leftarrow s_{max}$, return S to step 2, else
6. Return S as solution.

Figure 5.4 The hill-climb search algorithm



Search continues until the best feature in the current row is not better than that of the previous row.

Figure 5.5 Hill climb search for the feature space with 12 features. Evaluation function is the performance of the T x 25 X 12 MLP neural network, where T is the number of features in current feature space.

5.4.2 Results Using Wrapper Approach

Since our feature space was manageably small, all subsets in the hill-climb search space were examined, thus effectively eliminating the local performance maximum problem for this dataset. For each feature subset, a new MLP was trained with the experimentally generated training data set consisting of 30 patterns; the network was then tested on the remaining 54 patterns. The classification performance of the MLP was used as the evaluation function. The network architecture was T x 25 x 12, where T was the number of features in the current feature subset that was being evaluated. The same training and testing data sets were used for each and every network. On a 166 MHz machine with 32 MB RAM, the program took about 5 hours to complete, and on a 266 MHz machine with 64 MB RAM, the program converged in about 3.5 hours.

The feature subset that had the best performance with the least number of features on the model neural network architecture (4x25x12 with 30 training data, 54 testing data) was **PIB, OV275, SG and PDPP**. Although another subset with an additional feature had slightly higher performance, the four-feature subset was preferred because of its smaller dimension. To avoid a rapid growth in adding features, a subroutine was added to the algorithm to penalize marginally when adding an additional feature. It is interesting to note that PIB, OV275, and PDPP were also on the most successful coatings list of Zeller *et al.* [22], which was obtained through an exhaustive search.

The classification performance of this set with the blindly chosen training data was around 92%. A new training data of 30 patterns was randomly selected from the entire database, making sure that every class was represented at least once. The network with the same architecture (4x25x6) was initialized and trained with the new training data set, and tested

with the remaining 54 patterns. This feature subset classified all patterns correctly. The distribution of the training data and the performance are given in Table 5.3. This network was also tested with the expanded data set, and all but three of the patterns (all ethanol) of the total 180 patterns were classified correctly, giving a classification performance of 98.3%.

Table 5.3 Performance of the best feature subset as chosen by hill-climb

VOC	Train	Perf.		VOC	Train	Perf.
AC	2	7/7		**TCA**	2	7/7
MEK	1	7/7		**TCE**	2	7/7
ET	2	7/7		**HX**	3	7/7
ME	3	7/7		**OC**	4	7/7
ACN	2	7/7		**TL**	4	7/7
DCA	3	7/7		**XL**	2	7/7

5.5 Improving Wrapper Approach

Hill-climb has a couple of drawbacks that are addressed in this section. First, the hill-climb search technique is not necessarily a robust search technique. It is prone to being trapped in a local maximum in the performance space. Furthermore, when starting with one feature, the initial steps are more likely to result in the network not converging, since only one (or few) features will not be sufficient for convergence to the desired error minimum. Second, this search has a very high time complexity since it requires training and evaluation of a number of MLPs. However, it is interesting to note that the time required for evaluating a feature subset with a given number of features decreases as the number of features increases. In other words, finding the best subset with one feature takes more time than finding

the best subset with six features, given that the best five features are known from previous iterations.

When the total number of features is large, a more effective approach is to intelligently identify a few of the best possible features and then start the hill climbing approach from that point. This approach not only reduces the training time considerably, but it also prevents the convergence problems during the initial stages. Furthermore, knowing the first few critical features increases performance by avoiding any initial missteps starting the hill climb. We note that if no prior information is known about the data and/or importance of the possible features, statistical procedures can be used to determine features that are likely to be carrying important information.

One such procedure is using the variance of the features among different classes. Intuitively, features whose values change when the class changes carry more information than features whose values do not change with class. In addition, if the value of a particular feature is constant regardless of the class, then that feature provides no discriminatory information; therefore, it is of no use. On the other hand, this approach has a major flaw. If a particular feature changes in each case, then the variance of this feature would be very high, but it would not be useful for classification. A typical case is the medical database example given earlier. Care must therefore be taken when selecting the feature that has the maximum variance among different classes, but minimum variance among the patterns of the same class. Such a feature is a good candidate as the best feature for that classification problem. A good normalization scheme is also necessary for this approach to work.

When applied to the gas sensing database, the features that had the highest variance among different classes were PIB and OV275. These were also identified as the best features by the hill-climb search. However, computing their variances took much less time than using the hill-climb approach. For completeness, the following is the list of the features, in descending order of their variances:

1. **PIB**
2. **OV275**
3. **GRAP**
4. **PSB**
5. **PDPP**
6. **APZ**
7. **DEGA**
8. **PCP**
9. **PDSCO**
10. **PDS**
11. **SG**
12. **PDSOH**

It should be noted that SG, which was chosen by the hill-climb search, was at the bottom of the list. The obvious question that comes to mind, therefore, is how this approach alone would perform if the features were chosen from the top of this list. To answer this question, the top four coatings from this list were chosen and the standard network was trained again with 30 cases. The distribution of the training data and the results of this network are shown in Table 5.4.

With 15 misclassifications, these results prove that features should not be chosen on the basis of their variance only. However, choosing a few features with the highest variance, and using them as initial features in the hill-climb search may provide the best of two worlds by reducing the total processing time of the search algorithm.

Table 5.4 The performance of the coatings chosen based on their variance

VOC	Train	Perf.		VOC	Train	Perf.
AC	2	6/7		**TCA**	3	5/7
MEK	2	5/7		**TCE**	1	1/7
ET	2	7/7		**HX**	4	7/7
ME	2	6/7		**OC**	3	6/7
ACN	3	6/7		**TL**	2	5/7
DCA	3	7/7		**XL**	3	7/7

5.6 Conclusions

Two feature selection methods for the VOC identification problem were examined. The first approach, using a decision tree to determine the features carrying the most information and then training a neural network with these features performed well on both databases. The correct classification performance was 93% on the original experimentally generated data set, and 96% on the expanded data set. One major drawback of this scheme is the number of parameters that need to be optimized for various options of the decision tree-generating algorithm (e.g., pruning, cross-validation, boosting, etc). On the other side, decision trees are much faster to train than neural networks.

The second approach based on a hill-climb search of the feature space performed very well; the network trained with the four features selected classified all signals correctly. This approach, unfortunately, can be computationally prohibitive if the number of features exceeds a certain limit. One simple solution of using statistical characteristics of features has been proposed for reducing the amount of time required by this approach. For VOC identifi-

cation problems, hill climb can be safely used since the number of sensors used rarely exceeds twenty.

Note that the hill-climb approach does allow the user to prespecify the number of features desired. In addition, this approach requires fewer parameters to be optimized; however, the network architecture needs to be determined by the user.

CHAPTER 6

INCREMENTAL LEARNING

6.1 Motivation

The performance of a classification algorithm relies heavily on the availability of a comprehensive training dataset that is representative of all patterns that the classifier is likely to encounter. However, acquiring such a training dataset is typically expensive and time consuming, and in some cases, it may not even be feasible. When large volumes of data need to be collected, it is often more practical to acquire the dataset in batches. In such cases, the classification algorithm is expected to process small batches of data in an incremental mode. Ideally, such an algorithm should also be able to learn new information, adapt to evolution in knowledge as well as reinforce existing knowledge.

Learning from new data without losing prior knowledge is defined as incremental learning. Most existing classification algorithms do not allow incremental learning of new data. When new data become available, these algorithms have traditionally been re-initialized and retrained from scratch using a combination of old and new data, resulting in a loss of all previous training. This phenomenon is known as *catastrophic forgetting*, and it is prevalent in many automated signal classification algorithms, including the multilayer perceptron, radial basis function, probabilistic, wavelet, and Kohonen neural networks. Furthermore, if the old data is not available when new data arrive, incremental learning is impossible for such algorithms. Therefore, the development of a general approach for incremental learning of new data is of significant importance and a subject of active research interest.

Although recent years have witnessed an increasing need for incremental learning for automated classification systems, incremental learning has not been formally defined in the literature. Consequently, several versions of this problem have been addressed in the literature with varying degrees of complexity [135]. At one end of the spectrum, incremental learning is trivialized by allowing retraining with old data, without adding new classes. At the other end, an incremental learning algorithm is expected to learn in an on-line setting, where the learning is carried out on an instance-by-instance basis with some instances introducing new classes. Algorithms that are currently available for incremental learning typically fall somewhere in the middle of this spectrum. In this dissertation, an algorithm is considered as a truly incremental learning algorithm, if it does not forget what has been previously learned. Therefore, an incremental learning algorithm should not require access to old data. Given this requirement, several scenarios in signal classification can be constructed:

1. New data without new classes, with no access to previous datasets
2. New data with possibly new classes, with no access to previous datasets
    i. New data including instances from all previous classes, as well as the new class
    ii. New data including instances from only some of the previous classes, as well as the new class
    iii. New data including instances only from the new class.
3. New data with or without new classes, with the learner having access to some statistical information regarding the previous data, such as mean, variance, covariance matrix, etc.

Developing one algorithm that can handle all the issues listed above may not be feasible. However, a base approach may be able to address all of these cases with minor modifications.

As the primary original contribution of this research, Learn++, an incremental learning algorithm is introduced in this chapter. Learn ++ allows learning from new data, which may possibly include instances from previously unseen classes, without requiring access to old data. Before describing Learn++ in detail, related work on incremental learning by other researchers is discussed first.

6.2 Literature Survey: An Incremental Work on Incremental Learning

A word of caution is in order before discussing earlier work on incremental learning. This is because the phrase "incremental learning" has been used rather loosely with widely differing meanings in the pattern recognition and artificial learning literature. A number of papers refer to growing and pruning of classifier architecture as incremental learning. Notable examples include growing neural network architectures one node at a time [136, 137, 138].

Another group of algorithms that claim incremental learning are those that modify the weights of a neural network architecture by retraining, typically with misclassified signals. Although, these schemes are closer in meaning to the definition of incremental learning as defined in this study, they violate the major requirement since they forget previous learning and require access to old data.

For example, Vo [139] describes an incremental learning algorithm for time delay neural networks (ITDNN), which supplements the original TDNN architecture with the capability of learning a misclassified pattern in a single epoch by adding a dedicated hidden layer unit. In

order to correctly classify a misclassified sample, the algorithm first tries to adjust the weights and biases without affecting the overall performance simply by reintroducing the offending instance to the network. If this cannot be done, which is typically the case, the algorithm adds an extra hidden unit to perform *template matching* on the incorrectly classified instance. The activation pattern of the offending instance at the first hidden layer constitutes the template, and the weights for the additional unit are chosen such that the patterns close to this instance produce a high activation at the extra node, whereas those patterns that are not similar to the template pattern are deactivated at the hidden unit. Each hidden unit is connected to only one output node, representing the correct class for those instances that are similar to the misclassified instance. This procedure is repeated for all misclassified patterns that do not fit into one of the previously generated templates.

Hoya and Constantinides [140] describe an incremental learning scheme for GRNNs and related family of networks such as PNN and exact RBF networks. Since exact RBF networks and GRNNs are based on look-up tables, which store the entire training database, a misclassified test sample is simply added to the training dataset, and hence to the lookup table, which then gets correctly classified. Since GRNNs do not need iterative training, this algorithm can be implemented in an on-line learning setting.

It should be noted that both of these algorithms do not conform to the description of incremental learning given above, since they do not learn new patterns but rather previously misclassified patterns.

Higgins and Goodman [141] have suggested incremental learning with a rule based network, which actually has the same fundamental idea of the work described in [140]. The algorithm involves "growing" a network incrementally using the new data without requiring

old ones. According to this algorithm, each training instance is considered a parent rule, and any rule that is obtained by removing one feature from a parent rule is considered a child rule. Best rules are first identified by computing the information content of each rule with respect to the training data. For each rule, the information content of all child rules are also computed, and if the information content of the child rule is higher than that of the parent rule, the parent rule is replaced by the child rule with the largest information content. This scheme serves as the feature extraction module for the algorithm. Once the best set of rules is determined, a neural network is constructed from these rules. The input layer nodes of this network correspond to attributes (features) of the rules, the hidden layer constructs the antecedents of the rules as conjunction of attributes, and the output layer nodes correspond to class attributes. The network is not fully interconnected since not all attributes are used in all rules. This network is then trained using a gradient descent algorithm, such as backpropagation, in an online mode (one instance at a time). Each new rule adds one hidden layer node to the network, if not classified correctly by the existing network. It should be noted that this scheme is essentially a lookup table, since all rules are actually memorized in the hidden layer nodes.

Yamauchi and Ishii [142, 143] describe *incremental learning with retrieving interfered patterns (ILRI)*, which partially satisfies our description of incremental learning. ILRI is capable of learning new patterns by modifying the weights of an RBF network through gradient descent. However, the instances that are misclassified (interfered patterns) after the weight update are put into the training dataset of the new database, hence incremental learning by retrieving interfered patterns. The authors do suggest, however, removing the requirement of

storing misclassified old data by "approximately regenerating a pseudo instance" from the basis function at the hidden layer node that corresponds to the misclassified instance.

In 1992, Ramani [144] proposed a different solution to the incremental learning problem. His solution was capable of learning not only new data, but new classes as well. He first made the following assumption: in a properly trained network, the transfer function from the input layer to the hidden layer as coded by the weights should only depend on the structure of the problem, and not only on the training samples used. Therefore, he postulated that this transfer function should hold even if a new class is added to the classification problem. A consequence of this postulation is that the weights for the first layer, once properly trained for a c class problem, should not necessarily change for a $c+1$ class problem. The only change, he claimed, should come from the second layer weights. Therefore, he initially trained an MLP with a c class dataset, and then kept the first layer weights constant during training with a $c+1$ class dataset corresponding to the same classification problem. The second layer was trained only with additional class instances, where the training data for second layer consisted of activation of the hidden layer nodes (due to new data and old first layer weights) and the corresponding class information for the new class. One node was added to the output layer during training with new data to accommodate the $c+1^{st}$ class.

As a twist of history, the very same week (and quite possibly the very same day) in 1992, at the very same conference where Ramani introduced his *second layer only* incremental training algorithm, Carpenter and Grossberg unveiled Fuzzy ARTMAP, which arguably became one of the most popular learning paradigms after the multilayer perceptron [145, 146]. Among all algorithms discussed in this section, Fuzzy ARTMAP is the only algorithm that can be considered as a truly incremental learning algorithm since it is capable of learning

from new data and new classes in the absence of old data. Fuzzy ARTMAP is based on mapping two unsupervised clustering procedures implemented by two fuzzy ART modules, ART_a and ART_b. During unsupervised training, ART_a receives input patterns whereas ART_b receives correct class information. ART_a then generates a new cluster, corresponding to an instance, if and only if, the existing clusters activated by ART_a for the new instance cannot be mapped to correct class identified by ART_b. Essentially, ARTMAP generates new clusters (prototypes) for all possibly dissimilar looking patterns, where the measure of dissimilarity is controlled by a vigilance parameter. The number of clusters generated by ART_a is typically much larger than the total number of classes. Each one of these clusters is then mapped to one of the clusters generated by ART_b, the number of which is equal to the number of total classes. Fuzzy ARTMAP training continues until all training patterns are correctly classified, and therefore the error on training data is always zero. Fuzzy ARTMAP fits into the incremental learning scheme perfectly because it continually generates new clusters as new data become available (if new data look dissimilar to previously seen data). One disadvantage of Fuzzy ARTMAP, as reported by a number of researchers, is that it is very sensitive to the order of presentation of the training data. Fuzzy ARTMAP is also extremely sensitive to the selection of the vigilance parameter, and finding the correct value for the vigilance parameters can be quite difficult.

In 1993, Chen and Soo [147] introduced an incremental feed-forward neural network (IFFN). They first describe *adaptive learning*, which involves learning of one additional instance given a previously trained network. They update the weights for the additional instance in such a way that the influence of the weight update on previously trained instances is minimum, which is satisfied by minimizing a weight-sensitivity cost function. Considering

incremental learning as a continuous adaptive learning, the authors apply the same scheme to the incremental learning of a new stream of data on an instance-by-instance basis. However, the weight-sensitivity cost function requires that previous training instances of the *partially trained* network be known, which prevents the algorithm from learning in a truly incremental nature. In order to remove this shortcoming, the authors proposed generating pseudo-training instances from the responses of the partially learned network since the hidden layer nodes of the network realized Gaussian type radial basis functions.

Between 1994 and 1996, Fu *et al.* [148, 149, 150] introduced an incremental backpropagation learning network that is capable of learning new data in the absence of old data. However this system, also based on learning new instances through minor modification of current weights by putting a bound on weight modification, is not able to learn new classes. In their algorithm, first the weights are modified to learn the new instance, within the limits of a weight modification bound. If this could not be achieved, a new neuron was added to the hidden layer. Furthermore, a weight decay factor was also incorporated into the output connection of each new node so that the neurons that were not being reinforced would get deleted.

In 1998, Tontini introduced a slight modification of fuzzy ARTMAP to reduce the sensitivity of the ARTMAP to the order of presentation of the input patterns by replacing the ART_a module with an RBF network [151].

Within the past year, incremental learning has become even more popular. A few new approaches were proposed in 1999. Bruzzone *et al.* introduced another merger of formerly known classifiers. They used Isodata to generate new clusters as new data become available, followed by an RBF to combine the clusters through a set of weights to minimize the sum of

square error at the output [152]. Yet another merger of formerly known methods came from Hebert *et al.* [153] where they combined a self-organizing map (SOM) and an MLP to obtain *self organizing perceptron (SOP)*. The idea was to overcome the main drawback of the MLP as a classifier which can only generate unbounded regions, so that the input data located near each other in the feature space tend to activate the same group of neurons in the MLP's hidden layer. To do this, the authors trained a SOM and a MLP in parallel, and then combined the output of SOM with the hidden layer outputs of the MLP to control the output layer weights as dictated by the SOM. This system also does not allow incremental learning of new classes.

Vailaya and Jain followed a different route for incremental learning in Bayesian classification of images [154]. They used a Bayesian classifier as their base classifier. The class conditional probabilities were obtained form linear combination of Gaussians centered at q *codebook vectors* which themselves were extracted by a vector quantizer using the n training instances ($q<n$). Given the prior probabilities of the classes, the maximum a posteriori probabilities were then computed to determine the Bayes classification (the maximum likely class given the data instance). Incremental learning was obtained through updating the codebook vectors, adding new ones according to the new data, if necessary.

Finally, Vijakumar *et al.* recently proposed a substantially different approach for incremental learning based on reproducible kernel Hilbert spaces [155]. Their scheme is closely related to another recent scheme, namely, support vector machines [91, 92]. The scheme is very elegant in its theory; however, it requires significant expertise in kernel methods. The procedure first requires the selection of an optimal search space H in the Hilbert space in which all functions to be learned must reside. For the selected H space, reproducing kernel is

generated by constructing orthonormal bases in the selected space. A correlation operator, which is a measure of the a priori information on the distribution in the function space, is then estimated. This a priori knowledge is analogous to the Bayesian prior used in Bayesian estimators. The correlation operator is then used recursively to learn a new function from the previous one and the current sample. Naturally, this algorithm is geared towards the more general problem of function approximation. However, the nature of the function to be approximated must be a priori known for selecting a suitable H space [155].

Although most of the algorithms described above offer novel approaches, their true performance is yet unclear since the algorithms have been tested on only one (or two) dataset(s), with respect to which the algorithms have been optimized. The only exception to this is probably the Fuzzy ARTMAP, which has been used and tested in numerous research efforts, including this one. Fuzzy ARTMAP has been tested for the VOC database, as described in the results section, where its performance is compared to that of Learn++.

Furthermore, most of the above algorithms employ a specific classification algorithm or a network structure as a base classifier, and none of them provides a general solution to make any classifier an incremental learning algorithm. Also, in most cases claims of incremental learning were not tested nor documented (except ARTMAP).

Learn++, the incremental learning algorithm that is introduced in this chapter, is an intuitive algorithm which is very simple to use, and it can theoretically convert any classifier into an incremental learning algorithm (though it has only been tested on MLPs so far).

6.3 Ensemble of Classifiers and Learn++

Learn++ is inspired by Schapire's *adaptive boosting* (AdaBoost) algorithm, originally proposed for improving the accuracy of weak learning algorithms. In "Strength of weak learning" [156], Schapire showed that for a two class problem, a *weak learner* that almost always achieves high errors can be converted into a *strong learner* (also known as probably approximately correct - PAC learner) that almost always achieves arbitrarily low errors using a procedure called *boosting*. Both Learn++ and AdaBoost are based on generating an *ensemble of weak classifiers*, which are trained using various distributions of the training data and then combining the outputs (classification rules) of these classifiers through a majority voting scheme. In the context of machine learning, a classification rule generated by a classifier is referred to as a hypothesis; hence, they will be used interchangeably throughout the rest of this chapter.

Independently, Littlestone *et al.* developed the *weighted majority algorithm*, which assigns weights to different hypotheses based on an error criterion. Weighted hypotheses are then used to construct a compound hypothesis which was proved to perform better than any of the individual hypotheses [157]. They also showed that the error of the compound hypothesis is closely linked to the error bound of the best hypothesis. Schapire and Freund later developed AdaBoost.M1 extending boosting to multi-class learning problems and regression type problems [158, 159, 160]. Schapire *et al.* have continually improved their work on boosting with statistical theoretical analysis of the effectiveness of voting methods [161]. Recently, they have introduced an improved boosting algorithm that assigns confidences to predictions of Quinlan's decision tree algorithm. Their new boosting algorithm can also handle

multi-class / multi-label databases, where each instance may belong to more than one class [162].

Independent of Schapire and Freund, Breiman developed an algorithm very similar to boosting in nature. Breiman's *bagging*, short for *bootstrap aggregating*, is based on constructing ensembles of classifiers through continually retraining a base classifier with bootstrap replicates of the training database [163]. In other words, given a training dataset S of m samples, a new training dataset S' is obtained by uniformly drawing m samples with replacement from S. This is in contrast to AdaBoost where each training sample is given a weight based on the classification performance of the previous classifier.

Both boosting and bagging require weak classifiers as their base classification algorithms because both procedures take advantage of the so-called *instability* of the weak classifier. This instability causes the classifiers to construct sufficiently different decision surfaces for minor modifications in their training datasets. Both bagging and boosting have been used for constructing strong classifiers from weak classifiers, and they have been compared and tested against each other by several authors [164, 165].

The idea of generating an ensemble of classifiers is not new. A number of other researchers have also investigated the properties of combined classifiers. In fact, it was Wolpert who introduced the idea of combining hierarchical levels of classifiers, using a procedure called *stacked generalization* [166]. Kitler *et al.* analyzed error sensitivities of various voting and combination schemes [167], whereas Rangarajan *et al.* investigated the capacity of voting systems [168]. Ji and Ma proposed an alternative approach to AdaBoost for combining classifiers. Their approach generates simple perceptrons of random parameters and then combines the perceptron outputs using majority voting [169]. It should be noted that the idea of

generating an ensemble of classifiers through randomizing the internal parameters of a base classifier (rather than modifying its training set) was previously introduced by Ali and Pazzani [170, 171]. Obtaining time and space efficiency, as well as good performance levels were the main motivations of their approach. Ji and Ma also gave an excellent review of various methods for combining classifiers in [172]. Dietterich reviewed ensemble of classifiers with comparison to other types of learners, such as reinforcement learners and stochastic learners [173].

However, the steady increase in research efforts on combining classifiers has been mostly limited to improving the performance of classifiers. Learn++ has emerged as a result of investigating the feasibility of ensemble of classifiers for incremental learning.

Due to the strong connection between AdaBoost and Learn++, the former is described briefly in Section 6.5, immediately following the terminology and the background given in Section 6.4. The connection between using ensemble of classifiers and incremental learning is described in Section 6.6, followed by a detailed description of Learn++ in Section 6.7.

Unlike AdaBoost, which was designed to improve the performance of a classifier, Learn++ is designed to give supervised classification algorithms (in particular neural networks) incremental learning capability. Furthermore, Learn++ inherits all "performance improvement" capabilities of AdaBoost, and hence it can be used to improve the classification performance of a classifier as well. A theorem on the training error bound of Learn++ is given in Section 6.8, demonstrating its performance improvement capabilities. Section 6.9 presents the simulation results on various synthetic and real world benchmark databases as well as on the VOC identification database. Fuzzy ARTMAP performance on the same VOC database is also discussed in this section.

In Section 6.10, two variations of Learn++ are introduced. These two new versions of Learn++ make use of Mahalanobis distances between instances and previously used training datasets to dynamically assign weights to the hypotheses to be combined. Learn++ using Mahalanobis distances for combining hypotheses requires that mean and covariance matrices of previously used training dataset be available which typically take up much less space than the original data. In Section 6.11, results obtained using these versions of Learn++ are presented for the VOC database, as well as those for an equally challenging, but significantly larger database of A-scans obtained from submarine hull weld inspections.

An interesting property of Learn++ is then described in Section 6.12. This property allows the algorithm to predict the reliability of its own classification decision. Reliability analysis results are also given in this section. Conclusions and discussion are presented in Section 6.13 along with directions for future work.

6.4 Strong and Weak Learning

Consider an instance space $\mathcal{X}$, a concept class $\mathcal{C}=\{c: \mathcal{X} \rightarrow \{0,1\}\}$, a hypothesis space $\mathcal{H}=\{h:\mathcal{X}\rightarrow \{0,1\}\}$, and an arbitrary (not necessarily known, not even necessarily computable) probability distribution $\mathcal{D}$ over the instance space $\mathcal{X}$. In this setup, c is the true concept that we wish to learn, h is the approximation of the learner to the true concept c. Although the following definitions are for a two-class concept, they can be naturally generalized to the n-class concept. We assume that we have access to an oracle, which obtains a sample $x\in \mathcal{X}$, according to the distribution $\mathcal{D}$, labels it according to c, and outputs $\langle x, c(x)\rangle$. Both training and testing are performed using the examples provided by the oracle.

Definition PAC (Strong) learning: A concept class $\mathcal{C}$ defined over an instance space $\mathcal{X}$ is said to be potentially *PAC (probably approximately correct) learnable* using the hypothesis class $\mathcal{H}$ (which may or may not be the same as $\mathcal{C}$) if for all target concepts $c \in \mathcal{C}$, a consistent learner $\mathcal{L}$ is guaranteed to output a hypothesis $h \in \mathcal{H}$ with error less than $\varepsilon>0$ and probability at least $(1-\delta)$, $\delta>0$, after processing a finite number of examples, m, obtained according to $\mathcal{D}$. The learner $\mathcal{L}$ is then called a PAC learning algorithm, or a *strong learner* [174, 175].

Note that PAC learning imposes very stringent requirements on the learner $\mathcal{L}$, since $\mathcal{L}$ is required to learn *all* concepts within a concept class with arbitrarily low error $\varepsilon>0$ (approximately correct) and with an arbitrarily high probability $(1-\delta)$, $\delta>0$ (probably correct). Such a learner that satisfies these requirements may not be realizable, and hence such a learner is only a *potentially* PAC learning algorithm. However, finding a learner $\mathcal{L}_0$ that can learn with fixed values of ε, (say ε_0) and δ, (say δ_0) might be quite conceivable.

Definition Weak learning: A concept class $\mathcal{C}$ defined over an instance space $\mathcal{X}$ is *weakly learnable* using the hypothesis class $\mathcal{H}$, if there exists a learning algorithm $\mathcal{L}_0$ and constants $\varepsilon_0<1/2$, and $\delta_0<1$ such that for every concept $c \in \mathcal{C}$ and for every distribution $\mathcal{D}$ on the instance space $\mathcal{X}$, the algorithm $\mathcal{L}_0$, given access to an example set drawn from $(c, \mathcal{D})$, returns a hypothesis $h \in \mathcal{H}$ with probability at least $1-\delta_0$ and $Error_{c,\mathcal{D}}(h) \leq \varepsilon_0$ [174, 175].

Note that unlike strong learning, weak learning imposes the least possible stringent conditions, since it is required to perform only slightly better than chance (for a two class problem), and only some of the time. We then ask the following question: If we have access to a weak learner of mediocre performance, can we convert it into a strong learner of good per-

formance? Surprisingly, the answer is a very enthusiastic *yes* in the strongest sense. Despite the significant difference in the performance requirements of the two learners, Shapire showed that weak learning and strong learning are equivalent, and devised *boosting* to convert a weak learner into a strong learner [158].

6.5 Boosting the Accuracy of a Weak Learner

Boosting is based on running the weak learning algorithm a number of times to obtain many weak hypotheses, and using a majority vote to determine the final hypothesis whose error is less than any one of the individual weak hypotheses. For the generation of each additional hypothesis, the learner is presented with a different distribution of the training data, and it is forced to learn increasingly difficult examples.

6.5.1 Boosting for Two-class Problems

Let c be a Boolean target concept and $\mathcal{D}_1=\mathcal{D}$, where $\mathcal{D}$ is the original distribution of the training data. We run the weak learning algorithm $\mathcal{L}_0$ with training examples from $\mathcal{D}_1$ and obtain the weak hypothesis h_1 such that $Error_{c,\mathcal{D}_1}(h_1)=\varepsilon_1<\varepsilon<½$. Attention is then focused on examples misclassified by h_1. A new set of training examples is obtained from a new distribution $\mathcal{D}_2$ as follows: An oracle flips a fair coin; on heads, it returns an example $<x, c(x)>$ such that $h_1 \neq c(x)$. On tails the oracle returns an example $<x, c(x)>$ such that $h_1=c(x)$. Therefore, the new distribution $\mathcal{D}_2$ picks up correctly classified examples with probability ½ and picks up misclassified examples with probability ½. Now let h_2 be the new hypothesis returned by the learner $\mathcal{L}_0$ on $\mathcal{D}_2$. This hypothesis will also have an

$Error_{\mathcal{L}, \mathcal{D}2}(h_2)=\varepsilon_2<\varepsilon_1<½$. The next hypothesis h_3 is then returned by $\mathcal{L}_0$ in which h_1 and h_2 disagree. h_3 will then have $Error_{\mathcal{L}, \mathcal{D}3}(h_3)=\varepsilon_3<\varepsilon_1<½$. Then the final hypothesis h is chosen from the majority voting of the three hypotheses. Shapire showed that $Error_{\mathcal{L}, \mathcal{D}}(h)$ is bounded by $3\varepsilon^2-2\varepsilon^3$, which is less than ε. That is, with each iteration, the error of the final hypothesis decreases and can potentially converge to an arbitrarily low value of error. Furthermore, he showed that it only takes polynomial time for the error to reach arbitrarily low values, and gave an upper bound on the number of training examples required to reach these low error levels [156, 175].

6.5.2 Boosting for Multiclass Problems: AdaBoost.M1

AdaBoost is based on the belief that a large number of solvers, each solving a simple problem, can be used to solve a very complicated problem when the solutions to simple problems are combined in an appropriate form.

AdaBoost.M1 [159, 160], Schapire and Freund's first extension to the original boosting algorithm, was developed to boost the performance of a multi-class weak learning classifier by generating various weak classification hypotheses and combining them through weighted majority voting of the classes predicted by the individual hypotheses. These hypotheses are obtained by retraining the classifier using a different subset of the training dataset, chosen strategically based on the performance of the previous hypothesis. In general terms, each instance in the training database is assigned a weight, and these weights are updated based on the performance of the previous hypothesis. Misclassified instances are assigned larger weights, whereas correctly classified instances are assigned smaller weights. The training dataset for the next hypothesis is then chosen based on the current weights of the instances.

Instances with higher weights have higher chances of being selected into the next training set. Furthermore, the weights are normalized to satisfy the conditions to form a probability distribution function, referred to as the *distribution* $\mathcal{D}$ of the training dataset. Consistently misclassified instances are considered as *hard* examples of the dataset, and the algorithm is designed to train subsequent classifiers with increasingly harder instances of the dataset.

Inputs to AdaBoost.M1 are

- sequence of labeled examples (training data, S) drawn randomly from an unknown distribution $\mathcal{D}$,
- weak learning algorithm, **WeakLearn,** and
- an integer T that specifies the number of hypotheses (iterations) to be generated by **WeakLearn.**

The algorithm AdaBoost.M1, which is given in Figure 6.1, proceeds as follows: In iteration $t=1,2,\ldots,T$, AdaBoost.M1 provides the weak learning algorithm, **WeakLearn,** with a training subset data drawn according to distribution D_t from the original training data $S=[(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)]$, where x_i are training data instances and y_i are the corresponding correct labels. **WeakLearn** then computes a hypothesis (classifier) h_t: X → Y, which correctly classifies a percentage of the training set. That is, **WeakLearn**'s goal is to find a hypothesis h_t, which minimizes the training error

$$\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i) \tag{6.1}$$

The initial distribution D_1 is typically chosen to be uniform over S, unless there is prior knowledge to choose otherwise, that is, $D_1(i) = 1/m$, $\forall i$. This gives equal probability to all instances in S to be drawn into the initial training data subset. The distribution is updated by

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t, & \text{if } h_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases} \tag{6.2}$$

where $Z_t = \sum_i D_t(i)$ is a normalization constant chosen to ensure that D_{t+1} will be a distribution, and $\beta_t = \varepsilon_t/(1-\varepsilon_t)$.

The parameter β can be thought of as a normalized error term, since for $0<\varepsilon_t<½$, $0<\beta<1$. In fact, Schapire *et al.* showed that

$$\beta_t = \frac{\varepsilon_t}{(1-\varepsilon_t)} \tag{6.3}$$

is the optimum choice for the parameter β [160].

The distribution update rule in Equation 6.2 ensures that weights for misclassified instances are increased, whereas weights for correctly classified instances are reduced. Thus, AdaBoost.M1 focuses on examples that seem to be hardest for **WeakLearn** to learn. At the end of T iterations, AdaBoost.M1 combines the weak hypotheses $h_1,\ldots,h_T$ into a single final hypothesis h_{final} by computing the weighted majority of the weak hypotheses as

$$h_{final}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log(1/\beta_t) \tag{6.4}$$

where weight of hypothesis h_t is defined to be log $(1/\beta_t)$ so that greater weight is given to a hypothesis with lower error. For a given instance x, Equation (6.4) outputs the label y, that maximizes the sum of the weights of the weak hypotheses predicting that label.

It should be noted that AdaBoost.M1 requires ε_t, the error of each hypothesis h_t, to be less than ½. For a binary class problem, this is the least restrictive requirement one could have, since an error of ½ for a binary class problem is equivalent to random guessing. Note that

Algorithm AdaBoost.M1

Input:

- Sequence of m examples $S=[(x_1, y_1),(x_2, y_2),...,(x_m, y_m)]$ with labels $y_i \in Y = \{1,...,C\}$ drawn from a distribution $\mathcal{D}$,
- Weak learning algorithm **WeakLearn,**
- Integer T specifying number of iterations.

Initialize $D_1(i) = \frac{1}{m}, \forall i$.

Do for $t = 1,2,...,T$:

1. Call **WeakLearn**, providing it with the distribution D_t.
2. Get back a hypothesis $h_t : X \rightarrow Y$
3. Calculate the error of h_t : $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$

   If $\varepsilon_t > ½$, then set $T = t - 1$ and **abort loop**.
4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$.
5. Update distribution D_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t , & \text{if } h_t(x_i) = y_i \\ 1 , & \text{otherwise} \end{cases}$$

   where $Z_t = \sum_i D_t(i)$ is a normalization constant chosen so that

   D_{t+1} becomes a distribution function

Output the final hypothesis:

$$h_{final}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$$

Figure 6.1 AdaBoost.M1

any hypothesis with an error larger than ½ can be negated to obtain an alternate hypothesis with an error less than ½. However, obtaining a maximum error of ½ becomes increasingly difficult as the number of classes increases, since for a k class problem the error for random guessing is $(k-1)/k$. Therefore, the choice of a weak learning algorithm with a classification performance of at least 50% may not be very easy.

Any classification algorithm can be substituted as a weak learner by modifying appropriate parameters. For example, a MLP with a larger number of nodes/layers and a smaller error goal is, in general, a stronger learner than the one with smaller number of nodes and a higher error goal. It should be noted that the use of strong learners that achieve high classification performance on a particular training data are not recommended for use with boosting since there is little to be gained from their combination, and/or they may lead to over fitting of the data [160,169]. One of the nice properties of the AdaBoost.M1 algorithm is that it is less likely to encounter over fitting problems since only a portion of the instance space is learned by individual hypotheses. In addition, an ensemble of weak learners performs at least as well as a strong learner, but in considerably less time, since strong learners spend most of the training time during fine-tuning at lower error rates. A conceptual representation of combining classifiers is illustrated in Figure 6.2. In this figure, the dark curve is the decision boundary to be learned. Individual classifiers (hypotheses) are illustrated with simple geometric figures, and each decides whether a point in the feature space is within or outside the decision boundary. Each simple shaped region of different shade represents the region learned by a weak learner.

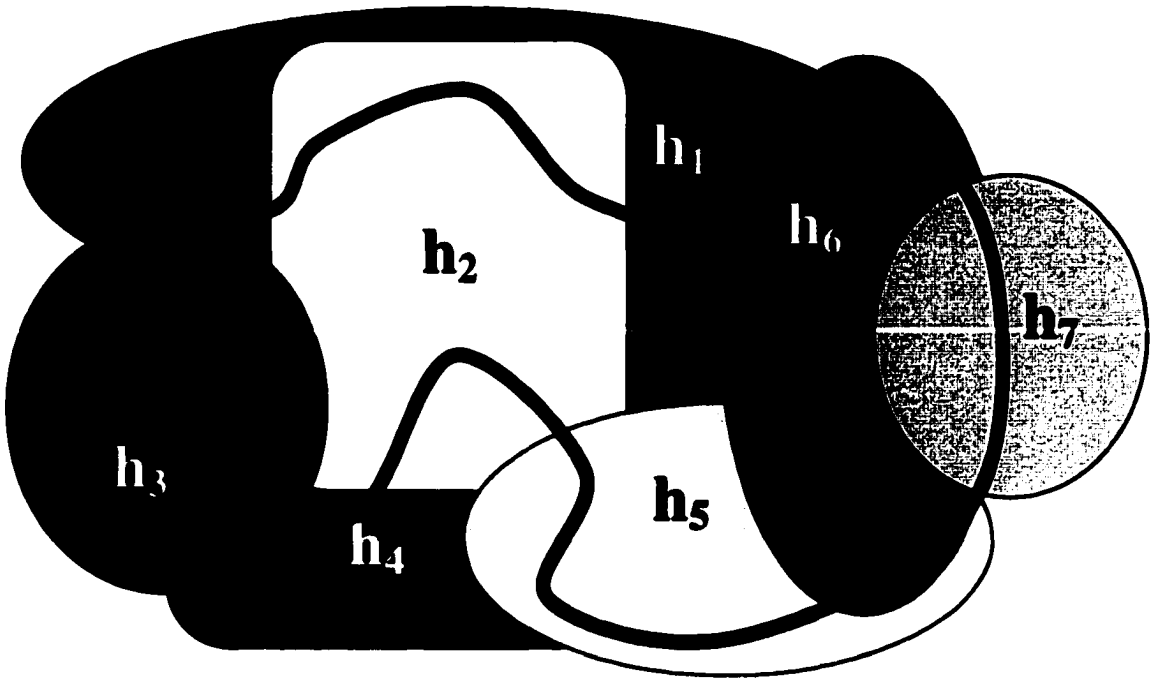


Figure 6.2 Conceptual representation of combining classifiers

Schapire and Freund have also developed AdaBoost.M2 for weak classifiers that are unable to obtain the 50% minimum performance requirement, as well as AdaBoost.R for boosting regression type learning problems [160].

6.6 Connection to Incremental Learning

In order to achieve incremental learning we assume that the new dataset S_{new} belongs to a slightly or significantly different portion of the original data distribution (data space) $\mathcal{D}$. In boosting, classifiers are added to learn regions of the pattern space that include increasingly "difficult" instances. Since, S_{new} is likely to be misclassified by the learner, instances of S_{new} can be considered to come from a "difficult to classify" region of the data distribution $\mathcal{D}$. Figure 6.3 conceptually illustrates the procedure of combining simple classifiers, similar to

Figure 6.2, but this time in the context of incremental learning. The dark curve is the decision boundary to be learned and the two sides of the dashed line represent the two training data S_1 and S_2. Individual classifiers (hypotheses) are illustrated with simple geometric figures, where $\mathbf{h_1}$ through $\mathbf{h_4}$ are generated due to training with S_1 and $\mathbf{h_5}$ through $\mathbf{h_8}$ are generated due to training with S_2. Each hypothesis decides whether a data point is within or outside the decision boundary, where simple shapes represent the region learned by a weak learner. Note that this setup is identical to that used by AdaBoost.M1, and hence Adaboost.M1 can be used for incremental learning of new data, with the understanding that new data corresponds to "harder" examples of the distribution.

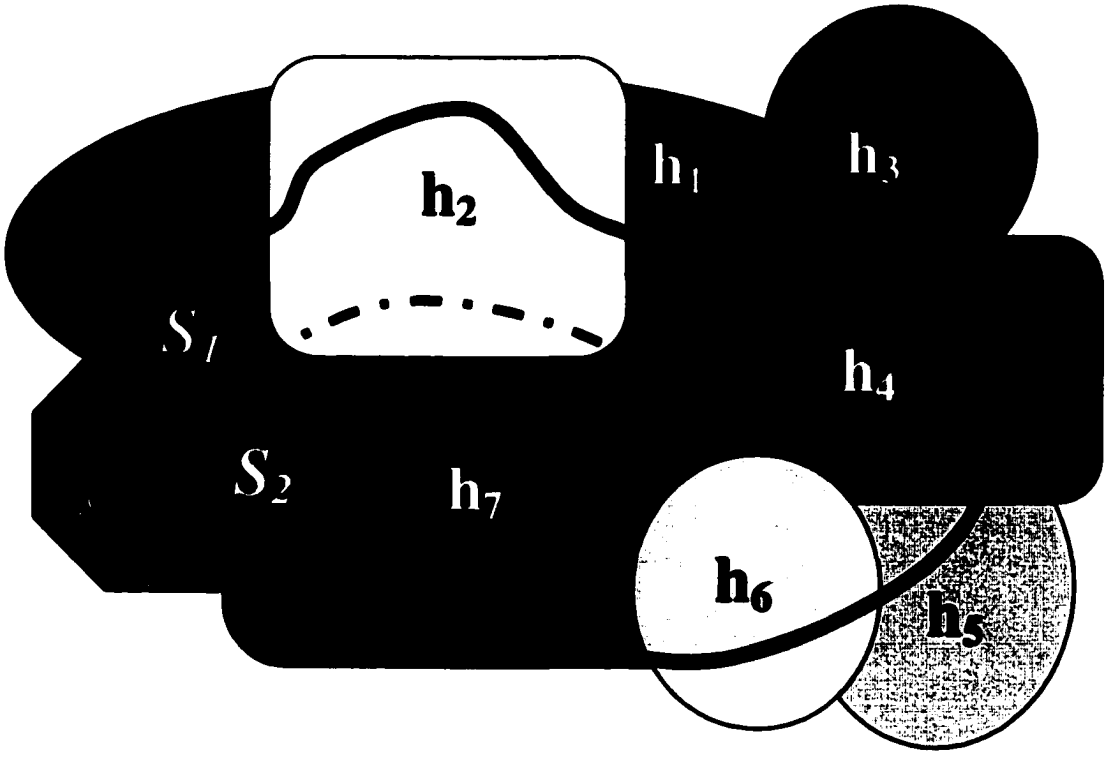


Figure 6.3 Conceptual representation of combining classifiers for incremental learning

However, the distribution update rule given in Equation 6.2 does not allow efficient incremental learning, particularly when new data include new classes. This is because the distribution update rule for D_{t+1} depends on the classification performance of h_t, a single hypothesis.

To understand the shortcoming of this distribution update scheme with respect to incremental learning, consider T hypotheses, h_1, h_2, ..., h_T generated with training datasets S_1, S_2, ..., S_T, all drawn from the same distribution $\mathcal{D}_1$, consisting of C classes. Assume that a new database of distribution $\mathcal{D}_2$ become available which includes instances from an additional $(C+1)^{st}$ class. AdaBoost.M1 will select the next training set S_{T+1} from $\mathcal{D}_2$ based on the classification performance of h_T, which was generated from a database that did not include the $(C+1)^{st}$ class. Furthermore, note that once the training set is selected, each classifier is independent and is likely to perform equally well (or poorly) on all classes (unless one class is particularly more difficult than others). In other words, hypothesis h_{T+1} will perform equally well on instances coming from $(C+1)^{st}$ class. Therefore, patterns from the $(C+1)^{st}$ class will not necessarily be selected into S_{T+1}, and they will have no advantage of being selected into the training dataset in the next few iterations. Consequently, learners will not be forced to focus on the patterns of the new class. The final weighted majority will then fail to recognize samples from the new class for many iterations to come, increasing the time and space complexity of the algorithm[1].

---

[1] Theoretically, if allowed to continue infinite number of times, AdaBoost.M1 should eventually be able to learn the new class. However, in all simulations where new classes were introduced, AdaBoost.M1 was unable to converge for a very long time (after which it was aborted) for all databases on which it was simulated.

The distribution update rule can, however, be forced to focus on instances of the new class, if the update rule is based on the combined performance of all t hypotheses generated during the previous t iterations. Let us call the weighted majority voting of the previous t hypotheses the *composite hypothesis* H_t. Note that when instances from a new class become available, they will be misclassified by H_t, since none of the previous t training sessions have seen instances from the new class. Therefore, updating the training dataset distribution based on the classification results of H_t will ensure that the selection of instances from the new class is favored.

The Learn++ algorithm, which incorporates these ideas into a smarter distribution update rule is described in the following section. As shown in the following sections, Learn++ not only allows the weak learning algorithm to learn incrementally from new data, but at the same time it converts the weak learning algorithm into a very powerful classifier.

6.7 Learn++: An Incremental Learning Algorithm

Learn++ is an algorithm that allows any classifier to learn incrementally from additional data, without forgetting what is previously learned, even when the new data includes a new class. To achieve this rather ambitious task, Learn++ introduces a number of modifications to the basic ideas of AdaBoost. First, the *training error* is redefined. In AdaBoost.M1, the error ε_t is the training error of the weak learner, calculated using the training patterns misclassified by h_t. This constitutes a problem, when using neural network type classifiers such as MLPs as base classifiers, since a converged neural network almost always performs close to 100% on its training data for any nontrivial error goal. This is particularly true for RBF, PNN, GRNN, ARTMAP type algorithms, since these algorithms guarantee 100% correct classifica-

tion on their training database. In order to ensure weak learning and a nonzero ε_t, Learn++ first divides the selected training dataset T_t into subsets TR_t and TE_t, where TR_t is the training subset and TE_t is the testing subset for the current training dataset T_t. During the t^{th} iteration, the weak learner is trained on TR_t, and tested on the entire set $T_t = TR_t + TE_t$. For each iteration, different training and testing subsets are selected based on previous performance. The error of t^{th} hypothesis on the combined ($TR_t + TE_t$) set is defined as ε_t. Eventually (almost) all patterns in the original training dataset will be seen by the weak learner, and hence using this definition of training error is justified.

Figure 6.4 presents the Learn++ algorithm. Initially all instances have equal likelihood to be selected into the first training dataset (unless there is prior knowledge to choose otherwise). In the following discussion, new *databases* that become available for incremental learning are denoted with the subscript **k** and the (unknown) distribution from which the $\mathbf{k}^{th}$ database is drawn will be denoted by the script $\mathcal{D}_k$, whereas the distribution of the current training *dataset* at t^{th} iteration is denoted by $\boldsymbol{D}_t$.

In each iteration t, the distribution $\boldsymbol{D}_t$ is obtained by normalizing the current weights of the instances in step 1. In step 2, training (TR_t) and testing (TE_t) subsets are randomly generated from the current database $\mathcal{D}_k$ according to the distribution $\boldsymbol{D}_t$. These subsets are used as inputs to **WeakLearn** in step 3, which returns the hypothesis h_t in step 4. The error, ε_t, is then computed from the misclassified patterns of $TR_t + TE_t$. If $\varepsilon_t > \frac{1}{2}$, h_t is discarded, and new TR_t and TE_t are generated. Instead of updating the distribution based on instances misclassified by h_t, Learn++ then calls the weighted majority voting in step 5 to compute the composite hypothesis, H_t.

Algorithm Learn++ (with major differences from AdaBoost.M1 indicated by ←)

Input: For each database drawn from $\mathcal{D}_k$ $k=1,2,\ldots,K$ ←

- Sequence of m training examples $S=[(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)]$.
- Weak learning algorithm **WeakLearn**.
- Integer T_k, specifying the number of iterations.

Do for $k=1,2,\ldots,K$:

Initialize $w_1^i = D(i) = 1/m$, $\forall i$, unless there is prior knowledge to select otherwise.

Do for $t = 1,2,\ldots,T_k$:

1. Set $\boldsymbol{D}_t = \mathbf{w}_t \Big/ \sum_{i=1}^{m} w_t(i)$ so that $\boldsymbol{D}_t$ is a distribution.
2. Randomly choose training TR_t and testing TE_t subsets according to $\boldsymbol{D}_t$. ←
3. Call **WeakLearn**, providing it with TR_t
4. Get back a hypothesis $h_t : \mathrm{X} \rightarrow \mathrm{Y}$, and calculate the error of h_t : $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ on

   $T_t = TR_t + TE_t$. If $\varepsilon_t > 1/2$, set $t = t - 1$, discard h_t and go to step 2. Otherwise, compute normalized error as $\beta_t = \varepsilon_t/(1-\varepsilon_t)$.
5. Call weighted majority, obtain the overall hypothesis $H_t = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log(1/\beta_t)$,

   and compute the overall error $E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i)\left[\left| H_t(x_i) \neq y_i \right|\right]$

   If $E_t > 1/2$, set $t = t - 1$, discard H_t and go to step 2. ←
6. Set $B_t = E_t/(1-E_t)$, and update the weights of the instances:

$$w_{t+1}(i) = w_t(i) \times \begin{cases} B_t, & \text{if } H_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases} \quad \leftarrow$$

$$= w_t(i) \times B_t^{1-\left[\left|H_t(x_i)\neq y_i\right|\right]}$$

Call Weighted majority on combined hypotheses H_t and **Output** the final hypothesis:

$$H_{final} = \arg\max_{y\in Y} \sum_{k=1}^{K} \sum_{t:H_t(x)=y} \log\frac{1}{B_t} \quad \leftarrow$$

Figure 6.4 Algorithm Learn++

Note that the composite hypothesis H_t is computed similar to the final hypothesis h_{final} in AdaBoost.M1, that is,

$$H_t = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t} \tag{6.5}$$

which makes the training error

$$E_t = \sum_{i:H_t(x_i) \neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i)[\,|\, H_t(x_i) \neq y_i \,|\,] \tag{6.6}$$

on misclassified instances, where $[\,|\bullet|\,]$ is 1 if the predicate holds true, and 0 otherwise. From this error, we compute the normalized error

$$B_t = \frac{E_t}{(1 - E_t)} \tag{6.7}$$

In step 6, the composite hypothesis, H_t, and its normalized error, B_t, are then used to update the distribution of the instances to be used in the next training session. The update rule is

$$\begin{aligned} w_{t+1}(i) &= w_t(i) \times \begin{cases} B_t, & \text{if } H_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases} \\ &= w_t(i) \times B_t^{\,1-[|H_t(x_i) \neq y_i|]} \end{aligned} \tag{6.8}$$

where $w_t(i)$ is simply the weight of the i^{th} instance for the t^{th} training session. At the end of T iterations (for each database $\mathcal{D}_k$), the final hypothesis is obtained by combining the composite hypotheses H_t.

$$H_{final} = \arg\max_{y \in Y} \sum_{k=1}^{K} \sum_{t:H_t(x)=y} \log \frac{1}{B_t} \tag{6.9}$$

In this algorithm, when a new dataset contains new classes, the composite hypothesis H_t will misclassify instances from the new class, and the algorithm will be forced to learn these instances.

A disadvantage of this approach is the large storage capacity required to store all hypotheses generated to learn the additional class. Addition of data with new classes results in generating a large number of hypotheses in order to remove the bias of the combined classifier towards the previous classes. This bias can be reduced significantly by changing the final classification rule to

$$H_{final} = \arg\max_{y \in Y} \sum_{k=1}^{K} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t} \tag{6.10}$$

which combines the original weak hypotheses h_t, rather than the combined hypotheses. It should be noted however that subsequent hypotheses are still generated with training data selected according to a distribution based on the performance of the composite hypotheses H_t, which, along with other modifications, distinguishes Learn++ from AdaBoost.M1. Experimental results, summarized in Section 6.9, demonstrate that both final classification rules given by equations 6.9 and 6.10 achieve the same performance level in incremental learning problems including new classes, whereas AdaBoost.M1 was unable to achieve desired incremental learning performances.

6.8 Theoretical Error Analysis of Learn++

In this section a theoretical error analysis of Learn++ algorithm is given, where the upper error bound of Learn++ on the training data is derived.

Theorem: *The training error of the Learn++ algorithm given in Figure 6.3 is bounded above by* $E \le 2^T \prod_{t=1}^{T} \sqrt{E_t \cdot (1-E_t)}$, *where* E_t *is also bounded above by the AdaBoost.M1 error bound* $E_t \le 2^t \prod_{s=1}^{t} \sqrt{\varepsilon_s \cdot (1-\varepsilon_s)}$.

Proof: Following a similar approach given in [160], we first show that the above error bound holds for a two class problem, and then show that a multi class problem can be reduced to a binary class problem, allowing the same error bound to hold for the multi class case as well. Let Learn+ represent the algorithm for binary problems.

In a binary class setting where the two possible values for y are 0 and 1, the equations for error terms and distribution update rules given in Figure 6.4 can be simplified as follows: The combined hypothesis is obtained by

$$H_t(x) = \begin{cases} 1, & \text{if } \sum_{t=1}^{T} \log(1/\beta_t) \cdot h_t(x) \ge \frac{1}{2} \sum_{t=1}^{T} \log(1/\beta_t) \\ 0, & \text{otherwise} \end{cases} \tag{6.11}$$

and the error for H_t is

$$E_t = \sum_{i:H_t(x_i) \neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i) \left| H_t(x_i) - y_i \right| \tag{6.12}$$

The distribution update rule is given by

$$\begin{aligned} w_{t+1}(i) &= w_t(i) \times \begin{cases} B_t, & \text{if } H_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases} \\ &= w_t(i) \times B_t^{1-\left| H_t(x_i) - y_i \right|} \end{aligned} \tag{6.13}$$

and the final classification rule for each dataset is

$$H_{final}(x) = \arg\max_{y \in Y} \sum_{t:H_t(x)=y} \log\frac{1}{B_t} = \begin{cases} 1, & \text{if } \sum_{t=1}^{T} \log(1/B_t) \cdot H_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} \log(1/B_t) \\ 0, & \text{otherwise} \end{cases} \quad (6.14)$$

We define the error of the final hypothesis as sum of the initial weights of the misclassified instances, that is,

$$E = \sum_{i:H_{final}(x) \neq y_i} D(i) \quad (6.15)$$

To find an upper bound for E, we analyze the final weights of the instances after T iterations, and associate these weights with the errors committed by composite hypotheses H_t. Note that after T rounds, the final weight for any instance is

$$w_{T+1}(i) = w_1(i) \cdot \prod_{t=1}^{T} B_t^{1-|H_t - y_i|} = D(i) \cdot \prod_{t=1}^{T} B_t^{1-|H_t - y_i|} \quad (6.16)$$

The summation over all instances gives

$$\sum_{i=1}^{m} w_{T+1}(i) = \sum_{i=1}^{m} D(i) \cdot \prod_{t=1}^{T} B_t^{1-|H_t - y_i|} \quad (6.17)$$

Comparing the sum of weights of all instances to the sum of the weights that are misclassified,

$$\sum_{i=1}^{m} w_{T+1}(i) \geq \sum_{i:H_{final}(x) \neq y_i}^{m} w_{T+1}(i) = \sum_{i:H_{final}(x) \neq y_i}^{m} D(i) \cdot \prod_{t=1}^{T} B_t^{1-|H_t - y_i|} \quad (6.18)$$

We now note that the final hypothesis H_{final} will make a mistake on instance i if and only if

$$\sum_{t=1}^{T} \log B_t^{-|H_t(x_i) - y_i|} \geq \sum_{t=1}^{T} \log(B_t)^{-1/2} \quad (6.19)$$

or alternatively, if and only if,

$$\prod_{t=1}^{T} B_t^{-|H_t(x_i)-y_i|} \geq \prod_{t=1}^{T} (B_t)^{-1/2} \tag{6.20}$$

Incorporating Equation 6.20 into Equation 6.18 for misclassified instances, we obtain

$$\sum_{i=1}^{m} w_{T+1}(i) \geq \sum_{i:H_{final}(x)\neq y_i}^{m} D(i) \cdot \prod_{t=1}^{T} B_t \cdot B_t^{-|H_t-y_i|} \geq \sum_{i:H_{final}(x)\neq y_i}^{m} D(i) \cdot \prod_{t=1}^{T} B_t^{1/2} = E \cdot \prod_{t=1}^{T} B_t^{1/2} \tag{6.21}$$

Hence,

$$E \leq \frac{\sum_{i=1}^{m} w_{T+1}(i)}{\prod_{t=1}^{T} B_t^{1/2}} \tag{6.22}$$

giving us an upper bound for the error of the final hypothesis. However, this upper bound based on the weights of individual instances is of little use, since it is difficult to keep track of the weights of every instance used for each hypothesis. The sum of these weights can also be limited by an upper bound, based on the errors of each H_t. Recognizing that $B^{\gamma} \leq 1-(1-B)\cdot\gamma$ for $0<B<1$, and starting with the sum of the weights of all instances,

$$\sum_{i=1}^{m} w_{t+1}(i) = \sum_{i=1}^{m} w_t(i) \cdot B_t^{1-|H_t(x_i)-y_i|} \leq \sum_{i=1}^{m} w_t(i)\left(1-(1-B_t)\left(1-|H_t(x_i)-y_i|\right)\right) \tag{6.23}$$

We now define the intermediate variable $\Psi_t(i) = |H_t(x_i) - y_i|$ as the loss of the t^{th} hypothesis on instance i, then the total error of the t^{th} combined hypothesis is

$$\begin{aligned} E_t &= \sum_{i=1}^{m} D_t(i) \cdot |H_t(x_i) - y_i| \\ &= \sum_{i=1}^{m} D_t(i) \cdot \Psi_t(i) = \mathbf{D_t} \cdot \mathbf{\Psi_t} \end{aligned} \tag{6.24}$$

Furthermore, recall from step 1 of Learn++ algorithm that

$$\mathbf{D}_t = \mathbf{w}_t \Big/ \sum_{i=1}^{m} w_t(i) \tag{6.25}$$

Substituting equations 6.24and 6.25 into Equation 6.23,

$$\begin{aligned}
\sum_{i=1}^{m} w_{t+1}(i) &\le \sum_{i=1}^{m} w_t(i) - (1-B_t)\sum_{i=1}^{m} w_t(i)(1-\Psi_t(i)) \\
&\le \sum_{i=1}^{m} w_t(i) - (1-B_t)\left(\sum_{i=1}^{m} w_t(i) - \mathbf{w_t}\cdot\Psi_t\right) \quad \text{from (6.25), we obtain} \\
&\le \sum_{i=1}^{m} w_t(i) - (1-B_t)\left(\sum_{i=1}^{m} w_t(i) - \mathbf{D_t}\cdot\Psi_t\left(\sum_{i=1}^{m} w_t(i)\right)\right) \text{and from (6.24)} \\
&\le \sum_{i=1}^{m} w_t(i) - (1-B_t)\left(\sum_{i=1}^{m} w_t(i) - E_t\cdot\sum_{i=1}^{m} w_t(i)\right) \\
&\le \sum_{i=1}^{m} w_t(i) - (1-B_t)\sum_{i=1}^{m} w_t(i)(1-E_t) \\
&\le \sum_{i=1}^{m} w_t(i)(1-(1-B_t)(1-E_t))
\end{aligned} \tag{6.26}$$

After T iterations, we obtain

$$\sum_{i=1}^{m} w_{T+1}(i) \le \prod_{t=1}^{T} 1-(1-B_t)(1-E_t) \tag{6.27}$$

Substituting Equation 6.27 into Equation 6.23,

$$E \le \frac{\sum_{i=1}^{m} w_{T+1}(i)}{\prod_{t=1}^{T} B_t^{1/2}} \le \frac{\prod_{t=1}^{T} 1-(1-B_t)(1-E_t)}{\prod_{t=1}^{T} B_t^{1/2}} \quad \text{or,}$$

$$\mathrm{E} \le \prod_{t=1}^{T} \frac{1-(1-B_t)(1-E_t)}{B_t^{1/2}} \tag{6.28}$$

which gives us an upper bound on the training error in terms of the normalized error and the actual error of the combined hypotheses H_t. Note that no relationship has been assumed between E_t and B_t in this derivation. We now find the optimum value for B_t from Equation 6.28.

Since all terms in Equation 6.28 are positive, we can take the derivatives individually for each t.

$$\frac{\partial\left(\dfrac{1-(1-B_t)(1-E_t)}{B_t^{1/2}}\right)}{\partial B_t}=0 \Rightarrow B_t=\frac{E_t}{1-E_t} \tag{6.29}$$

Finally, substituting Equation 6.29 into Equation 6.28,

$$E \le 2^T \prod_{t=1}^{T} \sqrt{E_t(1-E_t)} \tag{6.30}$$

which is identical in form to that of AdaBoost, except the errors of individual hypotheses h_t are replaced by the errors of composite hypotheses H_t. Furthermore, since each composite hypothesis H_t is obtained from individual hypotheses h_t much like the final hypothesis is obtained from the composite hypotheses, an identical error analysis can be carried out for each H_t individually, which will then yield

$$E_t \le 2^t \prod_{s=1}^{t} \sqrt{\varepsilon_s \cdot (1-\varepsilon_s)} \tag{6.31}$$

as the error of H_t, which is identical to overall error of AdaBoost.M1.

So far, we have shown the error bound for the binary classification problem; however, it is easy to show that the same analysis holds for multi-class problems by establishing a one-to-one mapping between the binary class and multi-class problems. Again following a similar approach to that in [160] for each instance in the Learn++ training set (x_i, y_i), we define a Learn+ instance $(\tilde{x}_i, \tilde{y}_i)$ with $\tilde{x}_i$ = some random number, and $\tilde{y}_i = 0$. We also define the initial distribution for Learn+ instances to be the same as Learn++ instances.

For each iteration t we pass the hypothesis $\tilde{H}_t(i) = [\![H_t(x_i) \neq y_i]\!]$ as if **WeakLearn** returns it to Learn+. Note that according to this formulation, if Learn++ misclassifies x_i, then it will return 1 to $\tilde{H}_t(i)$. Since the correct class of the corresponding $\tilde{x}_i$ is zero (all instances for Learn+ are of class zero by our previous definition), then $\tilde{H}_t(i)$ misclassifies this instance as well. On the other hand, if Learn++ correctly classifies instance x_i, it will return 0 to $\tilde{H}_t(i)$, and since this is also the correct class for all Learn+ instances, $\tilde{H}_t(i)$ also classifies the corresponding instance $\tilde{x}_i$ correctly. In other words, when the multi-class algorithm makes an error, the binary class algorithm makes an error, and when the multi-class algorithm correctly classifies an instance, so does the binary class algorithm. Since initial distributions for both algorithms were defined to be identical, errors computed by both algorithms will also be identical, hence $\tilde{E}_t = E_t$, $\tilde{B}_t = B_t$, and $\tilde{\mathbf{w}}_t = \mathbf{w}_t$. Therefore, the error of the final hypothesis E will also be identical to that given in Equation 6.30. ◆

6.9 Learn++ Performance Results

6.9.1 Simulation Databases

Five databases were used to test the incremental learning capabilities of the procedure described in the preceding section. These databases were chosen from a variety of sources to test the robustness of the algorithm.

1. *Vehicle* database: The vehicle database is a small database obtained from University of California at Irvine (UCI) Machine Learning Repository web site [176]. This dataset was specifically chosen because it is one of the most challenging datasets in the UCI repository. Typical performance levels reached by most learning algorithms have been in the

range of 60% to 70% for this data set. The database consists of 846 instances, each instance having 18 attributes, and belonging to one of four classes. This database was used to test the incremental learning algorithm with no new classes.

2. *Optical Digits*: Also obtained from the UCI repository, optical digits is a large database, consisting of a training set of 3823 instances and the test data set of 1797 instances. Only 1200 instances of the training dataset were used in this study. The characters were numbers, 0 through 9, and they were digitized on an 8x8 grid, creating 64 attributes as shown in Figure 6.5. Two of those attributes were zero for all instances, and hence were removed from the feature set. This database was also used to test for incremental learning with no new classes.
3. *Rectangular Regions*: This is a simple synthetic database of two attributes and four classes, artificially generated for testing for incremental learning with new classes. The data are plotted in Figure 6.6 in Section 6.9.4 where the performance of Learn++ on this dataset is discussed.
4. *Circular Regions*: This dataset, consists of five concentric circles in a two dimensional space, where the five circular rings produced the five classes. This database was specifically generated to test the invariance of Learn++ to the order of presentation of data. The classifiers were originally trained with three classes, and the other two classes were added later. Figure 6.7, in Section 6.9.5, illustrates the data.
5. *VOC Database*: This is the VOC mixture data for the identification of the five dominant VOCs described in detail in Chapter 3. The classifiers were originally trained with using three dominant VOCs, and the other two were added later.

It should be noted that in all simulations, no old data were used in subsequent stages of learning, strictly complying with the notion of incremental learning. Furthermore, each simulation was tested on an independent data set that was never used during training at any stage. It should also be noted that the algorithm developed does not depend on a specific classifier. We have applied this algorithm by simulating **WeakLearn** algorithm using a MLP with a relatively large error goal. MLP was chosen in the following implementations, since it is the most commonly used learning algorithm for classification purposes. However, Learn++ is independent of the weak learner used. In the following discussions, *each hypothesis* refers to the decision surface generated by each MLP.

Results on each database are presented and discussed in the following sections.

6.9.2 Vehicle Data

The 846 instance database was divided into four subsets, $S1$ through $S3$ of 210 instances each for training, and *TEST* of 216 instances for validating the classification performance. Instances in *TEST* were never seen by any of the classifiers.

For each *training session*, only one of the training datasets was used. That is, only $S1$ was used during the first training session, only $S2$ was used during the second training session, and so on. For each training session k=1,2,3, thirty hypotheses were generated by Learn++ according to the algorithm given in Figure 6.4. Each hypothesis h_t (t=1,2,...,30) of the training session k was generated using a training subset TR_t and a testing subset TE_t, each with 120 instances drawn from S_k (k=1,2,3). A single layer MLP of 18 input nodes, 30 hidden layer nodes and 4 output nodes with an error goal of 0.1 has been used as the base classifier. The results are summarized in Table 6.1. Recall that during training with $S2$, $S1$ was not

shown to the algorithm; however, $S = S_1 \cup S_2$ was used to evaluate the performance of the algorithm *after the second training session*. Note that this evaluation is used for display purposes only, since $S = S1 \cup S2$ was never used by the algorithm for training. During training with $S3$, the merged data was $S = S1 \cup S2 \cup S3$, and so on. Note that for the first training session, $S=S1$.

In the second column, the average classification performances of individual hypotheses are given for each dataset, where classification performances are computed as the percentage of correctly classified test instances. For example, during the first training session, **Training 1**, the individual hypotheses (MLPs) had an average classification performance of 62% on $S1$. This is attributed mainly to the small training datasets, and lack of convergence due to high error goal. Since the MLP was used as a weak learner, this performance was perfectly adequate.

Increasing the number of hidden layer nodes, reducing the error goal, or increasing the number of training instances would improve the performance of individual MLPs; however, this would defeat the purpose of using weak classifiers.

Table 6.1 Classification performance of Learn++ on vehicle database

Inc. Learn ++ ↓ Dataset	Average perf. per learner	Training 1 (30)	Training 2 (30)	Training 3 (30)
S_1	62%	93%	82%	79%
S_2	60%	-	86%	78%
S_3	64%	-	-	91%
S	62%	93%	84%	82.6%
TEST	**57%**	**78%**	**80.4%**	**83%**

Despite the 62% classification performance of individual hypotheses on the average on *S1*, Learn++ had a 93% classification performance on the same set by using an ensemble of 30 such hypotheses, demonstrating its performance improvement capabilities similar to that of AdaBoost.M1.

The fourth column shows the results of **Training 2** for *S2*, which generated additional 30 hypotheses. The weighted majority of the hypotheses generated in **Training 1** and **Training2** had classification performances of 82% on *S1*, and 86% on *S2*, giving an average of 84% on $S = S1 \cup S2$. The performance of these 60 hypotheses on the test set improved to 80.4% from 78%.

Finally, the last column shows the results of **Training 3** for *S3*, which also generated 30 hypotheses. The weighted majority of these 90 hypotheses provided classification performances of 79% on *S1*, 78% on *S2*, and 91% on *S3*, giving an average of 82.6% on $S = S1 \cup S2 \cup S3$. The performance on the test set improved to 83% after *S3* was shown to Learn++. The gradual increase in the performance of Learn++ on the *TEST* set, as new data are introduced, demonstrates the incremental learning capability of the algorithm, without forgetting previously learned information.

A similar terminology is used in the following paragraphs in presenting the results on other databases. Furthermore, in order to distinguish performances of individual hypotheses from those of ensemble of hypotheses through weighted majority, the latter will be referred to as *Learn++ performance* through out the rest of this chapter.

6.9.3 Optical Digits Database

A total of 1200 instances containing all ten classes were randomly selected from 3823 instances comprising the original training dataset. These 1200 instances were divided into six subsets $S1$ through $S6$ to construct six datasets of 200 instances each. For each training session k=1,2,...,6, 30 hypotheses were generated using the Learn++ algorithm given in Figure 6.4. Each hypothesis h_t (t=1,2,...,30) of the training session k was generated using a training subset TR_t and a testing subset TE_t, each with 100 instances drawn from S_k (k=1,2,...,6). An additional validation set, $TEST$, was used for validation purposes. Instances in $TEST$ were never seen by any of the classifiers. The classification algorithm used as **WeakLearn** was a single layer MLP of 62 input nodes, 30 hidden layer nodes and 10 output nodes with an error goal of 0.1. Figure 6.5 illustrates typical samples from the optical digits database. Note that this is a fairly noisy database due to quantization errors during discretizing, and the large variations in subjects' handwriting.

Table 6.2 presents the results, where the six training databases are denoted by $S1$ through $S6$. In the second column, the average performance of individual hypotheses is given for each dataset. The third column of the table shows the results at the end **Training 1**. Note that, although individual hypotheses had a classification performance of only 55% on their training sets, the weighted majority of these algorithms correctly classified 80%~90% of the training data, once again demonstrating the performance enhancement properties of Learn++ on any single dataset.

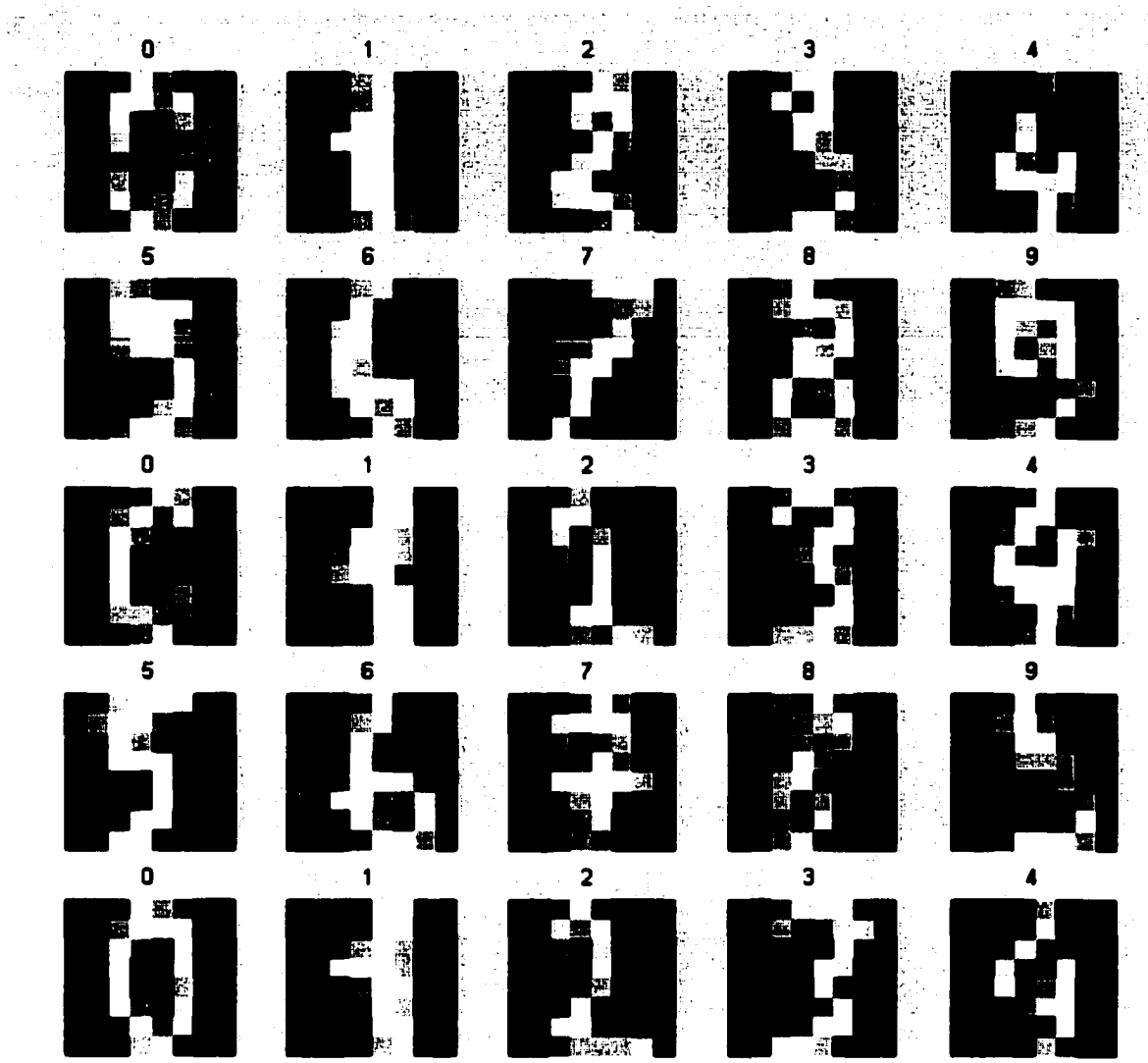


Figure 6.5 Optical digits database

The fourth column shows the results of **Training 2** with $S2$, which generated an additional 30 hypotheses. The weighted majority of the hypotheses generated in **Training 1** and **Training2** performed 94% on $S1$, and 93.5% on $S2$, giving an average of 93.7% on $S = S_1 \cup S_2$. The performance of these 60 hypotheses on the test set improved to 84.7% from 82%.

Table 6.2 Classification performance of Learn++ on optical digits database

Inc. Learn → ↓ Dataset	Average learner	Training 1	Training 2	Training 3	Training 4	Training 5	Training 6
S_1	55%	94%	94%	94%	93%	93%	93%
S_2	53%	---	93.5%	94%	94%	94%	93%
S_3	51%	---	---	95%	94%	94%	94%
S_4	53%	---	---	---	93.5%	94%	94%
S_5	56%	---	---	---	---	95%	95%
S_6	58%	---	---	---	---	---	95%
S	54.3%	94%	93.7%	94.3%	93.6%	94%	94%
TEST	**41.3%**	**82%**	**84.7%**	**89.7%**	**91.7%**	**92.2%**	**92.7%**

Similarly, the last column shows the results of **Training 6** with *S6*. The weighted majority of these 180 hypotheses had an average of 94% classification performance on $S = S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5 \cup S_6$. The classification performance on the validation set *TEST* was 92.7%, which improved steadily as we introduced new data. This demonstrated the incremental learning capabilities of Learn++ for a problem that did not include new classes.

6.9.4 Rectangular Regions Database

This database had four classes and consisted of four training datasets, *S1* through *S4*, and a validation dataset, *TEST*. *S1* and *S2* comprised of 366 and 394 instances, respectively, including instances from classes 1, 2 and 3. Only 200 instances were used during training, and the remaining instances (along with those used for training) were used for evaluating individual hypotheses. *S3* and *S4* included instances from all four classes and had 500 instances each.

Two hundred and fifty instances from these datasets were used for training. Figure 6.6 illustrates this dataset. The performance of Learn++ on this database is shown in Table 6.3.

Several points can be observed from the results in Table 6.3. First, since this was a very simple database, the boosting classification performances were all in the upper eighty to ninety percent ranges. However, our interests are mainly in the last row presenting the performance on the test dataset. Recall that test set includes instances from all classes. Since no class-4 instances were seen during the first two training sessions, the boosting classification performance on this data set was in lower 70%+ range. As the algorithm reached the third training session, instances from the fourth class also became available in the training set, and the boosting performance suddenly jumped to 94%. Addition of the fourth dataset provided only a minor improvement compared to that of the third set, increasing the boosting performance to 96%.

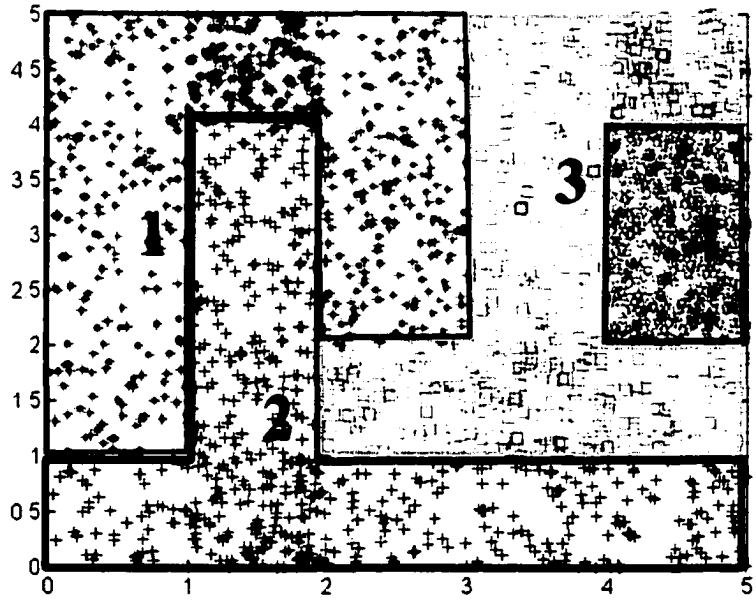


Figure 6.6 Rectangular regions dataset

Table 6.3 Classification performance of Learn++ on rectangular regions database

Inc. Learn → ↓ Dataset	Average learner	Training 1 (30)	Training 2 (30)	Training 3 (7)	Training 4 (4)
S_1	67%	99%	98%	86%	93%
S_2	72%		98%	86%	92%
S_3	68%			91%	94%
S_4	70%				96.4%
S	69%	99%	98%	87.6%	93.8%
TEST	**48%**	**71%**	**72.7%**	**94.2%**	**96%**

The second important point is the number of iterations that were required to reach these accuracy levels. The numbers in parentheses in the first row indicate the number of iterations (number of hypotheses generated) used during each training session. Note that only seven hypotheses were generated in **Training 3** and only four in **Training 4**. In fact, the performance rates started oscillating around the maximum performance levels after the indicated number of iterations. These observations imply that there is little or nothing to gain in allowing the algorithm to continue after that stage.

6.9.5 Circular Regions Database

Circular regions database is a synthetic database of concentric rings with two attributes and five classes. The database is artificially generated for testing the performance of Learn++ on incremental learning when instances with new classes are introduced. Figure 6.7 illustrates this database. In an attempt to see if the order of presentation has any effect on the performance of this algorithm, two sets of simulations were made on this database with six training datasets, *S*1 through *S*6 and a validation dataset. The order in which classes were intro-

duced to the algorithm was different in these cases. Table 6.4 summarizes the data distribution of this database. Note that in the first case, Learn ++ was initially trained with classes 1, 2 and 3. Classes 4 and 5 were added later in two separate training sessions. In the second case, Learn++ was initially trained with classes 1, 3 and 5, whereas classes 4 and 2 were added later. Total number of instances in each dataset and the number of instances used for each training session, are also shown in Table 6.4. The validation data *TEST* included instances from all classes for both cases.

Table 6.4 Data distributions for circular regions database

Database	S_1	S_2	S_3	S_4	S_5	S_6	*TEST*
Total number of instances	151	155	241	251	250	250	500
Classes included for 1st run	1,2,3	1,2,3	1,2,3,4	1,2,3,4	1,2,3,4,5	1,2,3,4,5	1,2,3,4,5
Classes included for 2nd run	1,3,5	1,3,5	1,3,4,5	1,3,4,5	1,2,3,4,5	1,2,3,4,5	1,2,3,4,5
Number of instances in training set	90	90	140	140	140	140	-

Table 6.5 summarizes the results obtained with this dataset. As expected, the Learn++ performance on the validation dataset *TEST* shows sudden jumps as instances of new classes become available during **Training 3** and **Training 5**. Also as expected, the improvement in the performance after **Training 4** and **Training 6** are minor compared to the previous sessions, since these sessions bring no instances with new classes. This also shows itself in the number of hypotheses generated during each training session (these numbers are given in the first row).

Table 6.5 also shows an additional column titled "Last 8", which indicates the Learn++ performance of the last eight hypotheses. Note that these hypotheses were trained with a dataset that included all classes, and one might expect that once these hypotheses are generated, earlier hypotheses are no longer necessary. As the last column in Table 6.5 illustrates, this is not true, since the last eight hypotheses alone were not adequate to give satisfactory performance. This demonstrates that all hypotheses are indeed necessary for the final classification.

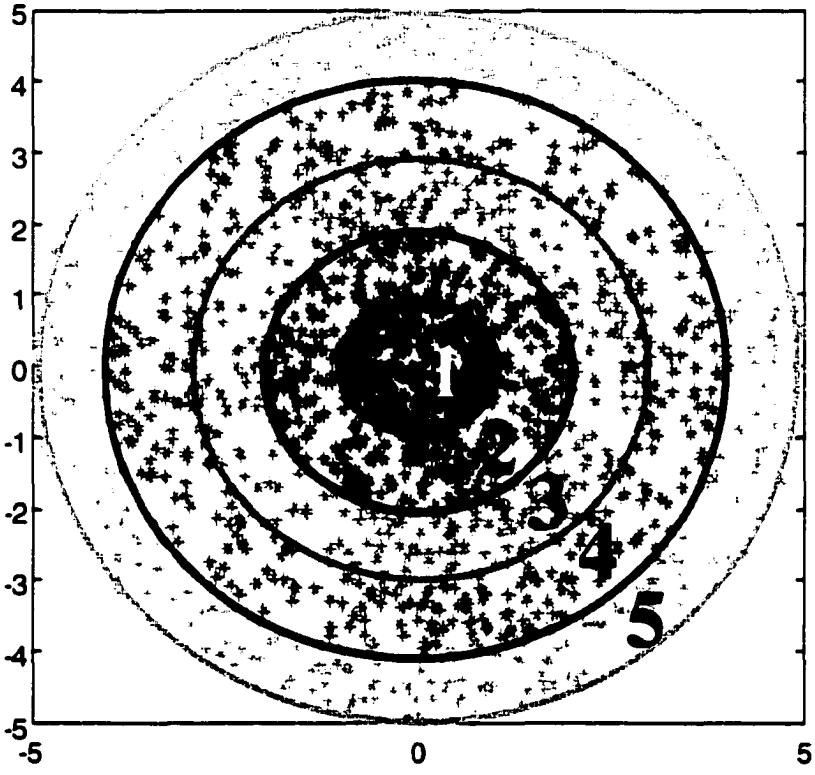


Figure 6.7 Circular regions database

Table 6.5 Classification performance of Learn++ on circular regions database (1st run)

Inc. Learn → ↓ Dataset	Training1 (10)	Training 2 (10)	Training 3 (15)	Training 4 (8)	Training 5 (15)	Training 6 (7)	Test S
S_1	98.7%	96.7%	91.4%	91.4%	95.3%	95.3%	41.7%
S_2	---	96.1%	87.1%	85.8%	92.2%	91.6%	40.6%
S_3	---	---	98.3%	98.3%	72%	90.8%	51.5%
S_4	---	---	---	93.6%	77%	88.4%	49.8%
S_5	---	---	---	---	88%	95.2%	60.4%
S_6	---	---	---	---	---	96.4%	53.6%
S	98.7%	96.4%	92.2%	92.2%	84.9%	92.9%	49.6%
TEST	**55.6%**	**56.8%**	**73.2%**	**74.4%**	**85.8%**	**89.6%**	52.8%

The results obtained with the second dataset are given in Table 6.6. Comparing results from Table 6.6 with the corresponding entries of Table 6.5, it can be concluded that the algorithm is invariant to the order of presentation of data when a MLP is used as the weak learning algorithm.

Another point of interest is the slight decline of Learn++ performance on the previous training datasets. This is probably due to the fact that the algorithm forces the subsequent weak learners to focus on the data that come from the new classes.

Once again we note that the performance on the validation data steadily increased as additional data became available, with largest jumps in the performances coming from the training sessions that introduced a new class. The incremental learning of new data without introducing a new class improves the classification performance only marginally since most of the knowledge to be learned was learned during the previous training.

Table 6.6 Classification performance of Learn++ on circular regions database (2nd run)

Inc. Learn. → / Dataset ↓	Training 1	Training 2	Training 3	Training 4	Training 5	Training 6
S_1	100%	100 %	94.8%	94.6%	91.1%	92.8%
S_2	---	100%	92.4%	95.2%	90.6%	88.5%
S_3	---	---	100%	96.8%	91.5%	93.5%
S_4	---	---	---	98.0%	90.5%	91.1%
S_5	---	---	---	---	83.0%	86.8%
S_6	---	---	---	---	---	88.8%
S	100 %	100 %	95.7%	96.1%	89.3%	90.3%
TEST	**59.4%**	**59.8%**	**72.2%**	**73.4%**	**80.8%**	**88.0%**

6.9.6 Mixture VOC Database

The final test was implemented on the real world data of the *NCT preprocessed* mixture VOC database, described in Chapters 3 and 4. The database consisted of 384 patterns, each with six attributes, belonging to one of five dominant VOC classes. The database was divided into four subsets, three for training and one for validation. The distributions of these datasets into five classes are given in Table 6.7. Note that as *S2* and *S3* were generated, the distribution was deliberately biased towards instances from the new classes, TCE, and xylene, respectively. The reason for doing so was simply to simulate a case where a new data would mostly be composed of signals from a new class. The network had a 6x30x5 architecture, with an error goal of 0.05, however, the classification performance of Learn++ was not too sensitive to these parameters. We were able to obtain similar results for a variety of network architectures and error goals.

Table 6.7 Data-class distribution for the VOC database

	ETHANOL	TCE	OCTANE	XYLENE	TOLUENE
S_1	**20**	**0**	**20**	**0**	**40**
S_2	5	**25**	5	**0**	5
S_3	5	5	5	**40**	5
TEST	34	34	34	40	62

Table 6.8 Classification performance of Learn++ on the mixture VOC data

Inc. Learn → ↓ Dataset	Training1	Training 2	Training 3
S_1	96.2%	77.5%	76.25%
S_2		87.5%	82.5%
S_3			90.0%
TEST	**60.78%**	**70.1%**	**88.2%**

Learn++ performances on these four sets of data are shown in Table 6.8. As observed in previous cases, the performance decreases in the first few training datasets, but increases significantly over the entire test set (which includes instances from all classes). For this particular database, we may have a physical reason for the performance decrease over the earlier training datasets. From our earlier experience, we know that the xylene patterns look remarkably similar to toluene and TCE patterns, which may account for some of the performance decrease after the last training session. However, a major factor is that the training data distributions were biased towards a certain class. Recall that the database was generated to be particularly biased towards instances from new classes, since in practice, it is possible for the new data to include only a few (or no) instances from previously learned classes.

It is also interesting to note that the classification performance of Learn++ on the VOC dataset is very comparable to those of various strong learners, developed in Chapter 4.

6.9.7 Fuzzy ARTMAP on VOC Database

As discussed above, Learn++ did consistently well on all databases, synthetic or real world, in learning new data which may include new classes. In fact, as shown in the VOC database example, it performed as well as a strong classifier that had access to the entire database. A more interesting, and fair, comparison is the performance of Learn++ with that of Fuzzy ARTMAP, an established algorithm for incremental learning.

Fuzzy ARTMAP was tested on the same VOC database, the distribution of which was given in Table 6.7. As discussed in the literature review section earlier in this chapter, Fuzzy ARTMAP is very sensitive to ρ_a, the vigilance parameter of the ART_a module. Therefore, various values were tried to find the optimum value of ρ_a to obtain the best performance of fuzzy ARTMAP. Table 6.9 presents the classification performance of Fuzzy ARTMAP for various values of ρ_a.

Table 6.9 Classification performance of Fuzzy ARTMAP on the mixture VOC data

Inc. Learn → ↓ Dataset	Training 1	Training 2	Training 3
S_1	100%	100%	100%
S_2		100%	100%
S_3			100%
TEST (ρ_a=0.85)	54.9%	68.1%	82.8%
TEST(ρ_a=0.90)	50.5%	67.2%	83.8%
TEST (ρ_a=0.95)	43.6%	59.3%	71.1%

Note that the classification performance of fuzzy ARTMAP is always 100% on training data, since according to the ARTMAP learning algorithm convergence is achieved only when all training data are correctly classified. Furthermore, once a pattern is learned, a particular cluster is assigned to it, and future training does not alter this clustering. Therefore, ARTMAP never forgets what it has seen as a training data instance. The improvement in the classification performance of the test data once again demonstrates that ARTMAP is indeed capable of incremental learning; however, even its best performance (83.8%) was not able to match that of Learn++ (88.2%). In particular, note that the performance of Learn++ was always better at each step of the training than that of Fuzzy ARTMAP. Also note that the slightest change in the vigilance parameter causes significant deterioration of the performance. Unlike ARTMAP, Learn++ is a very robust algorithm, since its parameters need not be fined tuned.

6.10 Learn++ with Mahalanobis Weighted Majority

Two issues of particular importance in the Learn++ algorithm are the distribution update rule and the weighted majority algorithm for combining the classifiers. In the Learn++ algorithm, the weights for combining the individual hypotheses are determined through the classification performances of these individual hypotheses on their own training data (care should be taken for not confusing the weights of training data instances with the weights of the hypotheses for the weighted majority).

As an alternate approach, assume that we knew which hypotheses are likely to classify a given instance correctly before assigning weights to the hypotheses. We could then weigh those hypotheses more heavily for the final classification of that instance. Such information

can be obtained by measuring the distance between each instance and the training datasets used to train each hypothesis. Unfortunately, this requires that we have access to the previously used datasets. However, computing the distance metrics does not require all the data, but rather certain statistical indicators of the data, such as the mean and the covariance matrix.

The number of instances in the training data is typically much larger than the dimensionality, N, of the dataset. Hence the storage required for, say the covariance matrix of size NxN is significantly less than the storage required for the entire data.

In particular, if the mean and the covariance matrix of each dataset are available, the Mahalanobis distance of an instance with each of the datasets can be computed by

$$M_t = (\mathbf{x} - \mathbf{m}_t)^T \mathbf{C}_t^{-1} (\mathbf{x} - \mathbf{m}_t) \tag{6.32}$$

where $\mathbf{x}$ is the unknown instance, $\mathbf{m}_t$ is the mean and $\mathbf{C}_t$ is the covariance matrix of TR_t, the training dataset used for the t^{th} hypothesis, and M_t is the Mahalanobis distance of $\mathbf{x}$ to TR_t. A smaller Mahalanobis distance indicates that the instance $\mathbf{x}$ is similar to instances that were in TR_t, and hence the t^{th} hypothesis is likely to classify this instance correctly. Therefore, the reciprocal of Mahalanobis distances can be used as weights of hypotheses. Note that the weight of hypothesis t changes with each instance, that is, the weights are updated dynamically.

In fact, this concept can be further improved by computing the Mahalanobis distances of each instance to subsets belonging to a particular class. If at iteration t, the classifier was trained with a dataset which included instances from C classes, we can partition TR_t into C subsets, TR_{tc} containing instances of TR_t belonging to class c, $c=1,2,...,C$.

We can then compute M_{tc} as

$$M_{tc} = (\mathbf{x} - \mathbf{m}_{tc})^T \mathbf{C}_{tc}^{-1} (\mathbf{x} - \mathbf{m}_{tc}) \qquad c = 1,2,\cdots,C \tag{6.33}$$

the Mahalanobis distance of $\mathbf{x}$ from TR_{tc}, where $\mathbf{m}_{tc}$ is the mean of TR_{tc}, and $\mathbf{C}_{tc}$ is the covariance matrix of TR_{tc}. The Mahalanobis weight of the t^{th} hypothesis can then be obtained as

$$MW_t = \frac{1}{\min(M_{tc})} \qquad c = 1,2,\cdots,C \tag{6.34}$$

where MW_t is dynamically updated for each data instance $\mathbf{x}$. This scheme finds the minimum Mahalanobis distance between instance $\mathbf{x}$ and each one of the C datasets TR_{tc}, and assigns the Mahalanobis weight of the t^{th} hypothesis as the reciprocal of this minimum Mahalanobis distance.

It can be argued that the classification decision is already being made by the choice of the minimum Mahalanobis distance, since if instance $\mathbf{x}$ belongs to a particular class, and instances from that class have been used in the current dataset, then the Mahalanobis distance between $\mathbf{x}$ and TR_{tc} is likely to be minimum among all others. This is indeed true for datasets with non-overlapping classes and instances, which do not have any noise. In practice, however, this is not the case, and a decision based on the Mahalanobis distance only would not achieve good classification performances on challenging datasets, such as the VOC dataset. Note that the Mahalanobis distance is not used directly for making a classification decision, but rather it is used to assign a weight to various hypotheses. Using this scheme, Learn++ simply tries to make a more intelligent distribution of weights among the hypotheses generated. The algorithm Learn++ using Mahalanobis weighted majority voting is given in Figure 6.8.

Note that normalized error terms, β_t of individual hypotheses or B_t of the composite hypotheses, are not used to determine the weights for combining the hypotheses. In fact, it is no

longer necessary to compute β_t; however, it is necessary to compute B_t, since the distribution update rule is still based on this normalized error term of composite hypotheses.

It should be noted that the Mahalanobis distance calculations require the computation of an inverse of a covariance matrix; therefore, this matrix must be ensured to be non-singular. The covariance matrix is usually a nonsingular matrix, as long as two instances are not repeated in the original dataset (that is all rows and columns are linearly independent), and the number of instances exceeds the dimensionality. These requirements ensure that the covariance matrix is a full rank matrix.

In general, the number of training data is significantly larger than N. However, since the training data is chosen randomly from a given distribution, even if the training instance selection is done without replacement, some instances may be too similar to each other, hence making the inverse covariance matrix close to singular.

The algorithm given in Figure 6.8 is designed to handle reasonable number of such cases. Note from Equation 6.34 that the weights assigned to hypotheses are inverses of the Mahalanobis distances. If a covariance matrix is singular, then the Mahalanobis distance is computed to be infinity, the reciprocal of which is zero. Since the maximum of weights is considered in step 5 of the algorithm, cases causing singularities are effectively discarded.

Simulation results in testing Learn++ using Mahalanobis distance to compute the weights of the hypotheses is discussed in the next section where the algorithm is applied to the VOC database, and an ultrasonic weld inspection database.

Algorithm Learn++ with Mahalanobis Weighted Majority Voting

Input: For each dataset drawn from $\mathcal{D}_k$ $k=1,2,\ldots,K$

- Sequence of m training examples $S=[(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)]$.
- Weak learning algorithm **WeakLearn** (MLP).
- Integer T_k, specifying the number of iterations.

Do for $k=1,2,\ldots,K$:

Initialize $w_1^i = D(i) = 1/m$, $\forall i$, unless there is prior knowledge to select otherwise.

Do for $t = 1,2,\ldots,T_k$:

1. Set $\mathbf{D}_t = \mathbf{w}_t \Big/ \sum_{i=1}^{m} w_t(i)$ so that $\mathbf{D}_t$ is a distribution.
2. Randomly choose training TR_t and testing TE_t subsets according to $\mathbf{D}_t$.
3. Call **WeakLearn**, providing it with TR_t
4. Get back a hypothesis $h_t : \mathbf{X} \rightarrow \mathbf{Y}$, and calculate the error of h_t : $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ on $TR_t + TE_t$. If $\varepsilon_t > 1/2$, set $t = t - 1$, discard h_t and go to step 2.
5. Compute the variances and means of the datasets used, and call Mahalanobis weighted majority, to obtain composite hypothesis $H_t = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} MW_t$, where MWt is the Mahalanobis weight of t^{th} hypothesis.
6. Compute the overall error $E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i)[\,|\,H_t(x_i) \neq y_i\,|\,]$

   If $E_t > 1/2$, set $t = t - 1$, discard H_t and go to step 2
7. Set $B_t = E_t/(1-E_t)$, and update the weights of the instances:

$$w_{t+1}(i) = w_t(i) \times \begin{cases} B_t, & \text{if } H_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases}$$
$$= w_t(i) \times B_t^{\,1-[|H_t(x_i)\neq y_i|]}$$

Call Mahalanobis weighted majority on all hypotheses generated so far and **Output** the final hypothesis: $H_{final} = \arg\max_{y\in Y} \sum_{k=1}^{K} \sum_{t:h_t(x)=y} MW_t$

Figure 6.8 Learn++ with Mahalanobis weighted majority

6.11 Classification Performance of Learn++ using Mahalanobis Distance

6.11.1 VOC Mixture Dataset

Learn++ with Mahalanobis weighted majority was first evaluated on the VOC dataset explained in Chapter 3 and Section 6.9.6 of this chapter. However, to ensure that no instance was chosen twice for any training subset, the distribution of the training data was slightly modified. Note that according to Table 6.7, there were 5 instances of ET, OC and TL in *S2*, and 5 instances of ET, OC, TL, and TCE in *S3*. Since the dataset is six dimensional, there had to be more than six instances from each class to ensure full rank of the covariance matrix. The distribution of the modified dataset is shown in Table 6.10.

Table 6.10 Data-class distribution for the VOC database

	ETHANOL	TCE	OCTANE	XYLENE	TOLUENE
S_1	**20**	0	**20**	0	**40**
S_2	10	**25**	10	0	10
S_3	10	15	10	**40**	10
TEST	24	24	24	40	52

Learn++ was tested on this dataset using two different versions of the Mahalanobis distance, namely the ones given in Equation 6.32 and Equation 6.33. In the first definition, we compute the Mahalanobis distance between each instance and the entire training set TR_t used to obtain the t^{th} hypothesis. In this case, we only need to ensure that each dataset had at least six instances, which is easily satisfied by both data distributions given in Table 6.7 and Table 6.10. Table 6.11 presents the classification results where the Mahalanobis distance was computed according to Equation 6.32.

Table 6.11 Classification performance of Learn++ on VOC data using Mahalanobis distance in combining classifiers (1st run)

Inc. Learn. → ↓ Dataset	Training 1 (5)	Training 2 (10)	Training 3 (5)
S_1	98.8%	86.3%	75.0%
S_2		89.9%	90.1%
S_3			94.1%
S	98.8%	87.4%	86.4%
TEST	**56.7%**	**64.0%**	**86.6%**

Compared to the results given in Table 6.8, Table 6.11 shows some deterioration in the overall performance on the test dataset, though the algorithm does demonstrate its incremental learning capabilities. Table 6.12 shows the classification results obtained using Mahalanobis distance as defined in Equation 6.33. The performance using this version of Mahalanobis distance is better than that using the previous version and that for the original Learn++ performance given in Table 6.8.

Table 6.12 Classification performance of Learn++ on VOC data using Mahalanobis distance in combining classifiers (2nd run)

Inc. Learn. → ↓ Dataset	Training 1 (7)	Training 2 (10)	Training 3 (16)
S_1	100%	97.5%	92.5%
S_2		96.4%	96.4%
S_3			92.9%
S	100.0%	97.0%	93.6%
TEST	**57.3%**	**67.1%**	**89.6%**

6.11.2 Ultrasonic Weld Inspection Database - Ascans

Learn++ using Mahalanobis distance in weighted majority voting (according to Equation 6.33) was also tested on a highly challenging database of overlapping classes. This database consisted of ultrasonic weld inspection signals (UWI). The problem is to identify the types of defects that are commonly encountered in weld inspection, namely, crack, lack of fusion (LOF), slag and porosity. Detailed information on this database can be found in [177]. The training dataset consisted of the ultrasonic weld inspection signals from these four classes. Each instance was obtained by taking 149 DWT coefficients of the 512-long time domain A-scans. The entire training set contained 974 crack, 2535 LOF, 1201 slag and 448 porosity signals. Figure 6.9 illustrates typical normalized A-scans obtained from this database.

Note that crack, LOF and slag signals look remarkably similar; only porosity signals show some difference in the high energy support region and ringing effects. Figure 6.10 shows the corresponding 149 DWT coefficients for the signals in Figure 6.9, where the data reduction is clearly noticeable (all samples beyond coefficient 150 were negligibly small).

Three distinct datasets *S1*~ *S3* were then generated, where *S1* had instances only from crack and LOF, *S2* had instances from crack, LOF and slag, and *S3* had instances from all four classes. A total of 2200 A-scans were randomly selected into these three datasets according to the distributions given in Table 6.13. A validation set, *TEST*, of 800 instances was also generated for evaluation purposes, and this dataset was never shown to the classifiers during training. The weak learner used to generate individual hypotheses was a single hidden layer MLP with 50 hidden layer nodes. The mean square error goals of all MLPs were preset to a value of 0.02 to prevent overfitting and to ensure a weak learning algorithm.

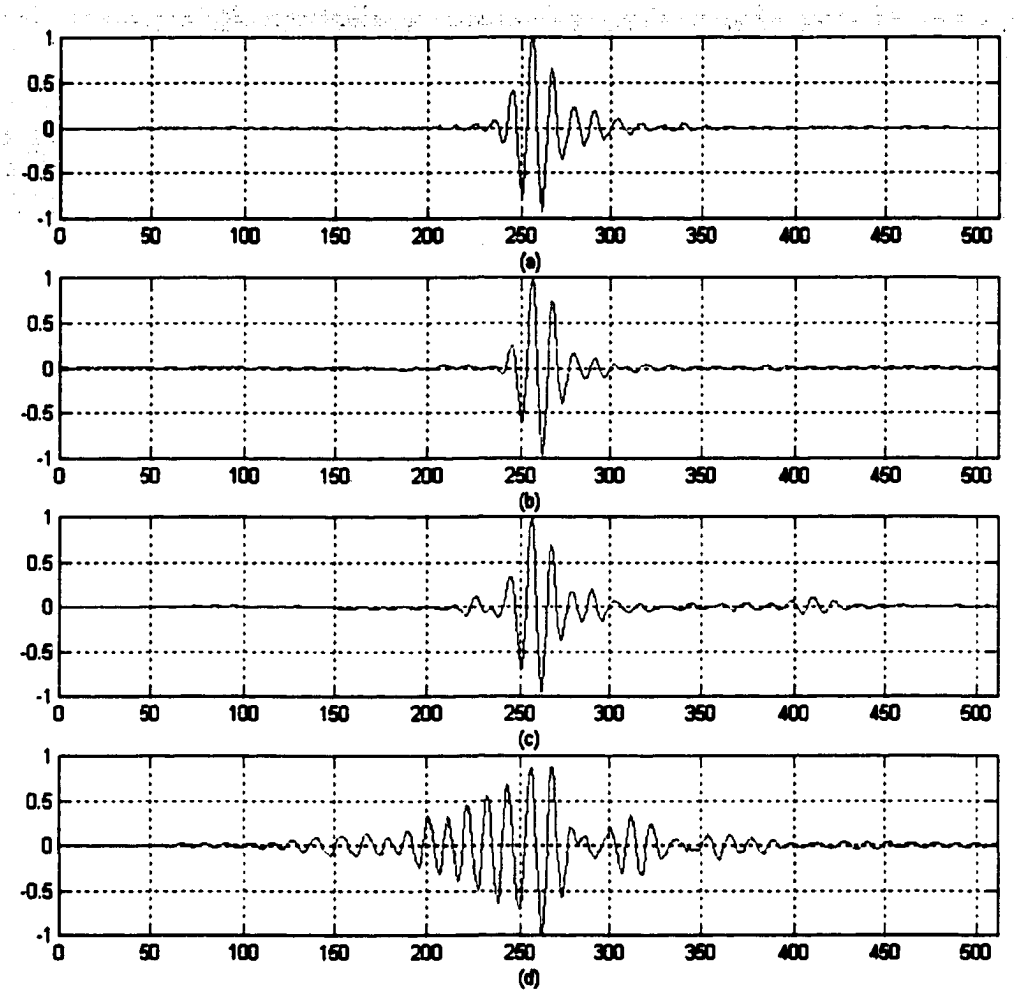


Figure 6.9 Typical A-scans (a) crack, (b) LOF, (c) slag, (d) porosity

Table 6.13 Distribution of weld inspection signals

	CRACK	LOF	SLAG	POROSITY
S1	300	300	0	0
S2	150	300	150	0
S3	200	250	250	300
TEST	200	300	200	100

Table 6.14 Classification performance of Learn++ using Mahalanobis distance for combining classifiers

Inc. Learn → ↓ Dataset	Training 1 (S1)	Training 2 (S2)	Training 3 (S3)
S_1	99.2%	89.2%	88.2%
S_2	-	86.5%	88.1%
S_3	-	-	96.4%
S	99.2%	87.5%	91.2%
TEST	**57.0%**	**70.5%**	**83.8%**

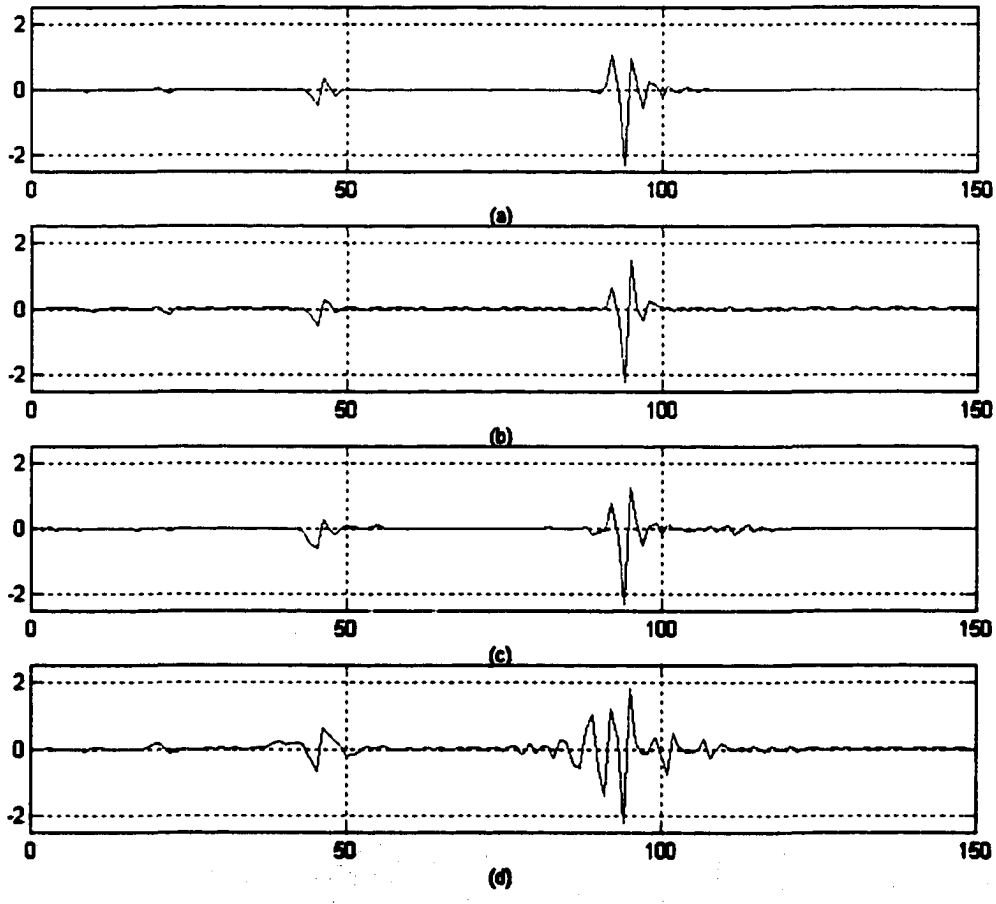


Figure 6.10 Corresponding DWT coefficients (a) crack, (b) LOF, (c) slag, (d) porosity

Table 6.14 presents the classification results of the algorithm on this data, illustrating a very familiar pattern of classification performance. This appears to be the signature of the Learn++ algorithm. As we have observed with other databases, Learn++ improved the performance on any given dataset from upper 50% levels (not shown in Table 6.14) of individual hypotheses to upper eighty and ninety percent levels. Also as in earlier cases, the performance on previous training datasets deteriorates slightly as new data are included, but the performance on the validation (*TEST*) data improves dramatically.

As a performance comparison, the same database was also used to train and test a single strong learner, a 149x40x12x4 two hidden layer MLP with an error goal of 0.001. The best test data classification performance of the strong learner has been around 75% [178], despite the fact that the strong learner was trained with instances from all classes.

6.11.3 Ultrasonic Weld Inspection Database - Cscans

The ultimate test for the Learn++ was given by testing its performance on UWI C-scan datasets, where each C-scan constituted of A-scans in a 3D volume. This dataset was divided into two parts: The first part, the training dataset, consisted of 109 C-scan images from which a total of 2200 A-scans were randomly selected and used for training Learn++ as described in Section 6.11.2. Not all images were used for training, however, since A-scans were randomly chosen. Furthermore, not all signals in this training set were used for training. In particular, those 800 signals used for testing were never seen by any of the networks. The second part of the C-scan dataset was the validation dataset. A-scans from this dataset were never seen by the networks. There were 50 C-scans in this database.

The procedure for classifying the C-scans was as follows: Along with the dataset, the exact locations of each flaw, as well as the type of the flaw, was known ahead of time, since each sample was previously hand scored using a combination of techniques, including ultrasonic and radiographic methods.

The flaw location on the C-scan image was carefully selected by a rectangular box cursor, and the A-scans which fell into this rectangular region were classified by Learn++ using the weighted majority voting, where the weights were determined according to Mahalanobis distance defined in Equation 6.33. A classification image was generated, based on the classification of each A-scan. The classification image was then post processed using a modified median filtering to remove isolated pixels, such as a single crack indication inside a large LOF area, or a very small porosity indication inside a very large slag area. Typically such indications are not common in practice (but do occur occasionally[2]), and median filtering is effectively used to remove isolated indications from an image. Table 6.15 summarizes the comparative results of Learn++ and a strong learner on training and validation datasets.

---

[2] Median filtering generally improves the visual interpretation of the result quite significantly. The only exception was for a porosity sample, in which the porosity region was known to be 0.1 inches long, roughly equal to the scanning resolution. It was quite encouraging to notice that Learn++ pinpointed this extremely small porosity indication inside a large crack/LOF region. Post processing obviously removed this isolated pixel, as shown in the corresponding C-scans for this sample in Figure A4.7. In all other cases, the post-processed classification at the indicated region was considered as the final classification.

Table 6.15 Comparison of Learn++ and strong learner on C-scans UWI data

	# of C-scans Training	# of C-scans Missed Unk / Training	Classification Performance	# of C-scans Validation	# of C-scans Missed Unk Validation	Classification Performance
Strong Learner	106	8/1	92.4 %	50	11/2	77.1%
Learn++	**106**	**1/0**	**99.1%**	**50**	**7/4**	**84.8%**

For C-scan classification, both Learn++ and the strong learner had samples that were classified as unknown. These refer to the cases where an equal number of A-scans from a given region had different classifications. It is also interesting to note that the only misclassification of Learn++ on the training data was in classifying an LOF a "slag" which was also called a slag by the strong learner. For the validation data, out of the 7 misclassified samples, 4 agreed with the classification of the strong learner. Appendix IV shows examples of original C-scan images and classification C-scan images obtained by Learn++.

6.12 Confidence of Learn++ in Its Decision

A classification algorithm that is capable of predicting its own reliability can be extremely valuable in evaluating the classification decisions. For example, decisions of false alarms can result in significant financial loss for industries since such decisions would require replacement of expensive undamaged components. Knowing the level of confidence in the classification decision would therefore be of paramount importance in industrial applications. Statistical analysis of classification results through hypothesis testing which uses the prior and post probabilities of expected outcomes can provide a good measure of the reliabil-

ity of the classification outcome, as shown by Ramuhalli [179]. However, these probabilities are often unknown and need to be estimated from the noisy data.

A very intuitive and straightforward alternative to estimating the reliability of the classification outcome is actually built in to the Learn++ algorithm. Recall that Learn++ is based on a weighted majority voting of multiple hypotheses trained with similar data. Therefore, the relative difference between the votes each class receives can be interpreted as how strongly Learn++ is confident about its decision. Essentially, if the majority of the (weighted) hypotheses agree on the class of a particular instance, we can interpret this outcome as a high confidence decision. If, on the other hand, the individual hypotheses votes are distributed equally among different classes, the final decision can be interpreted as a low confidence decision.

To formalize this approach, recall that the hypotheses are combined through

$$H_{final} = \arg\max_{y \in Y} \sum_{k=1}^{K} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t} \tag{6.35}$$

as described in Section 6.7, where $h_t(x)$ is the t^{th} hypothesis and $\log(1/\beta_t)$ is the weight of the t^{th} hypothesis. For Learn++ using Mahalanobis distance, hypotheses are combined by

$$H_{final} = \arg\max_{y \in Y} \sum_{k=1}^{K} \sum_{t:h_t(x)=y} MW_t \tag{6.36}$$

where MW_t is the Mahalanobis weight of the t^{th} hypothesis as described in Equation 6.33. For either of the combination schemes, let us assume that there are a total of T hypotheses generated in K training sessions for classifying instances into one of C classes. We can then define ξ_c, the total vote that class c receives, as

$$\xi_c = \sum_{t:h_t(x)=c} \log \frac{1}{\beta_t} \qquad \text{for original Learn++}$$

$$\xi_c = \sum_{t:h_t(x)=c} MW_t \qquad \text{for Learn++ with Mahalanobis}$$

$$t = 1,\cdots,T, \quad c = 1,\cdots,C \tag{6.37}$$

The final classification will then be the class for which ξ_c is maximum. Normalizing the votes received by each class

$$\xi_c = \frac{\xi_c}{\sum_{c=1}^{C} \xi_c} \tag{6.38}$$

allows us to interpret ξ_c as a measure of reliability of the decision on a 0 to 1 scale, with 1 corresponding to maximum reliability and 0 to no reliability. It should strictly be noted however that normalized ξ_c values do *not* represent the *accuracy* of the results, nor is it related to the statistical definition of confidence intervals determined through hypothesis testing. It is merely a measure of the confidence of the algorithm in its own decision, which we will call the reliability of the decision. Keeping this distinction in mind, we can heuristically define the following ranges:

$$\begin{aligned}
&0.0 < \xi_c < 0.4 \Rightarrow && \textit{Very low reliability} \\
&0.4 < \xi_c < 0.5 \Rightarrow && \textit{Low reliability} \\
&0.5 < \xi_c < 0.75 \Rightarrow && \textit{Medium reliability} \\
&0.75 < \xi_c < 1.0 \Rightarrow && \textit{High reliability}
\end{aligned} \tag{6.39}$$

A reliability analysis was performed for the mixture VOC database and the UWI database, based on the above described interpretation of the final voting results. Table 6.16 presents the reliability analysis of the original Learn++ classification on VOC mixture data.

Table 6.16 Learn++ confidence analysis on VOC mixture data

#	Ethanol Vote	Octane Vote	Toluene Vote	Xylene Vote	TCE Vote	LEARN++ Decision	Correct Class	Reliability
1	0.834	0.000	0.100	0.067	0.000	ETHANOL	ETHANOL	H
2	0.777	0.000	0.156	0.067	0.000	ETHANOL	ETHANOL	H
3	0.777	0.000	0.156	0.067	0.000	ETHANOL	ETHANOL	H
4	0.785	0.000	0.149	0.067	0.000	ETHANOL	ETHANOL	H
5	0.702	0.000	0.231	0.067	0.000	ETHANOL	ETHANOL	M
6	**0.309**	**0.000**	**0.483**	**0.160**	**0.048**	**TOLUENE**	**ETHANOL**	**VL**
7	0.834	0.000	0.084	0.067	0.016	ETHANOL	ETHANOL	H
8	0.000	0.822	0.074	0.082	0.022	OCTANE	OCTANE	H
9	0.000	0.751	0.145	0.082	0.022	OCTANE	OCTANE	H
10	0.000	0.725	0.145	0.108	0.022	OCTANE	OCTANE	M
11	0.000	0.751	0.145	0.082	0.022	OCTANE	OCTANE	H
12	0.000	0.751	0.145	0.082	0.022	OCTANE	OCTANE	H
13	**0.160**	**0.235**	**0.179**	**0.393**	**0.032**	**XYLENE**	**OCTANE**	**VL**
14	0.000	0.751	0.145	0.082	0.022	OCTANE	OCTANE	H
15	0.000	0.789	0.107	0.082	0.022	OCTANE	OCTANE	H
16	0.031	0.000	0.515	0.091	0.364	TOLUENE	TOLUENE	M
17	0.051	0.000	0.524	0.061	0.364	TOLUENE	TOLUENE	M
18	0.051	0.027	0.443	0.124	0.355	TOLUENE	TOLUENE	L
19	0.061	0.031	0.542	0.061	0.304	TOLUENE	TOLUENE	M
20	0.061	0.031	0.510	0.061	0.337	TOLUENE	TOLUENE	M
21	**0.031**	**0.027**	**0.243**	**0.302**	**0.397**	**TCE**	**TOLUENE**	**VL**
22	0.051	0.000	0.529	0.098	0.322	TOLUENE	TOLUENE	M
23	**0.000**	**0.751**	**0.145**	**0.082**	**0.022**	**OCTANE**	**TOLUENE**	**H**
24	**0.000**	**0.031**	**0.324**	**0.249**	**0.397**	**TCE**	**TOLUENE**	**VL**
25	**0.000**	**0.502**	**0.156**	**0.302**	**0.041**	**OCTANE**	**TOLUENE**	**M**
26	0.051	0.000	0.529	0.075	0.345	TOLUENE	TOLUENE	M
24	0.000	0.091	0.259	0.474	0.176	XYLENE	XYLENE	L
27	0.000	0.102	0.287	0.532	0.080	XYLENE	XYLENE	M
28	0.000	0.091	0.259	0.532	0.118	XYLENE	XYLENE	M
29	0.000	0.196	0.259	0.457	0.087	XYLENE	XYLENE	L
30	**0.355**	**0.000**	**0.184**	**0.239**	**0.222**	**ETHANOL**	**XYLENE**	**VL**
31	0.000	0.113	0.320	0.513	0.054	XYLENE	XYLENE	M
32	0.000	0.031	0.324	0.249	0.397	TCE	TCE	VL
33	**0.031**	**0.027**	**0.443**	**0.144**	**0.355**	**TOLUENE**	**TCE**	**L**
34	0.000	0.031	0.324	0.249	0.397	TCE	TCE	VL
35	0.000	0.058	0.270	0.275	0.397	TCE	TCE	VL
36	0.000	0.031	0.375	0.197	0.397	TCE	TCE	VL
37	**0.092**	**0.027**	**0.462**	**0.159**	**0.260**	**TOLUENE**	**TCE**	**L**
38	0.000	0.031	0.324	0.249	0.397	TCE	TCE	VL

Table 6.16 is a partial list of all instances in the TEST dataset explained in Table 6.7 in Section 6.9.6. In the first five columns, the vote each class received for each instance is provided as computed according to Equations 6.37 through 6.39. The Learn++ classification and the correct class are then given in the next two columns followed by the estimated reliability of the decision. In the last column, *H* is for high confidence, *M* is for medium confidence, *L* is for low confidence and *VL* is very low confidence, as determined according to Equation 6.39.

A number of interesting observations can be made from this table where all misclassified instances are indicated in bold. First, note that most misclassifications have low or very low confidences, with only one high and one medium confidence. The second interesting observation is that the confidence in correct classification tends to deteriorate towards the bottom of the table, which corresponds to instances of classes added during the incremental learning. Recall from Section 6.9.6 that xylene and TCE were added later to the training database during the second and third training sessions.

Table 6.17 presents similar information for Learn++ with Mahalanobis distance, computed according to Equation 6.32. Note that most misclassified instances still have low confidences, though there are a few more medium confidence misclassifications in this case. Also note that the deterioration in the correct classification confidence levels is considerably less severe when Mahalanobis distances are used for weighting the hypotheses. There were no correct classification with low confidence, and most correct classifications had high confidence. Considering that the weights of hypotheses were computed on an instance by instance basis, these results make intuitive sense.

Table 6.17 Learn++ with Mahalanobis confidence analysis on VOC mixture data (I)

#	Ethanol Vote	Octane Vote	Toluene Vote	Xylene Vote	TCE Vote	LEARN++ Decision	Correct Class	Reliability
1	1.000	0.000	0.000	0.000	0.000	ETHANOL	ETHANOL	H
2	1.000	0.000	0.000	0.000	0.000	ETHANOL	ETHANOL	H
3	0.874	0.000	0.126	0.000	0.000	ETHANOL	ETHANOL	H
4	1.000	0.000	0.000	0.000	0.000	ETHANOL	ETHANOL	H
5	1.000	0.000	0.000	0.000	0.000	ETHANOL	ETHANOL	H
6	0.940	0.000	0.060	0.000	0.000	ETHANOL	ETHANOL	H
7	**0.000**	**1.000**	**0.000**	**0.000**	**0.000**	**OCTANE**	**ETHANOL**	**H**
8	0.000	0.978	0.022	0.000	0.000	OCTANE	OCTANE	H
9	0.000	0.956	0.044	0.000	0.000	OCTANE	OCTANE	H
10	0.000	1.000	0.000	0.000	0.000	OCTANE	OCTANE	H
11	0.000	1.000	0.000	0.000	0.000	OCTANE	OCTANE	H
12	0.000	0.627	0.000	0.373	0.000	OCTANE	OCTANE	M
13	**0.000**	**0.166**	**0.280**	**0.168**	**0.386**	**TCE**	**OCTANE**	**VL**
14	0.000	0.775	0.000	0.225	0.000	OCTANE	OCTANE	H
15	**0.000**	**0.000**	**0.236**	**0.161**	**0.604**	**TCE**	**TOLUENE**	**M**
16	0.000	0.000	1.000	0.000	0.000	TOLUENE	TOLUENE	H
17	0.000	0.000	1.000	0.000	0.000	TOLUENE	TOLUENE	H
18	0.000	0.000	1.000	0.000	0.000	TOLUENE	TOLUENE	H
19	0.000	0.000	1.000	0.000	0.000	TOLUENE	TOLUENE	H
20	**0.000**	**0.000**	**0.054**	**0.468**	**0.478**	**TCE**	**TOLUENE**	**VL**
21	**0.569**	**0.000**	**0.431**	**0.000**	**0.000**	**ETHANOL**	**TOLUENE**	**M**
22	**0.000**	**0.578**	**0.108**	**0.314**	**0.000**	**OCTANE**	**TOLUENE**	**M**
23	0.000	0.000	1.000	0.000	0.000	TOLUENE	TOLUENE	H
24	**0.000**	**0.000**	**0.471**	**0.021**	**0.508**	**TCE**	**TOLUENE**	**M**
25	0.000	0.000	1.000	0.000	0.000	TOLUENE	TOLUENE	H
26	0.000	0.000	0.030	0.970	0.000	XYLENE	XYLENE	H
27	0.000	0.000	0.107	0.893	0.000	XYLENE	XYLENE	H
28	0.000	0.016	0.101	0.884	0.000	XYLENE	XYLENE	H
29	0.000	0.000	0.108	0.892	0.000	XYLENE	XYLENE	H
30	0.000	0.000	0.112	0.888	0.000	XYLENE	XYLENE	H
31	0.075	0.000	0.409	0.516	0.000	XYLENE	XYLENE	M
32	0.000	0.000	0.091	0.909	0.000	XYLENE	XYLENE	H
33	0.000	0.000	0.231	0.052	0.717	TCE	TCE	M
34	0.000	0.000	0.273	0.117	0.610	TCE	TCE	M
35	0.000	0.000	0.189	0.048	0.763	TCE	TCE	H
36	**0.000**	**0.000**	**0.580**	**0.000**	**0.420**	**TOLUENE**	**TCE**	**M**
37	0.000	0.000	0.205	0.047	0.748	TCE	TCE	M
38	0.000	0.000	0.257	0.107	0.636	TCE	TCE	M

Finally, Table 6.18 presents a similar confidence analysis table for the second version of Learn++ using Mahalanobis distance for combining classifiers, where Mahalanobis distance was computed as given in Equation 6.33. Although the performance of this algorithm was significantly better than that of the previous one (compare tables 6.11 and 6.12), there was not a significantly noticeable difference in the reliability levels.

However, one notable observation is that there is no longer a significant deterioration in the confidence levels of classification for the instances presented in the later stages of the training. Similar to the previous case, most correctly classified instances had high confidence, and most misclassified instances had low or medium confidences. Similar reliability measure analysis was also performed for the UWI database, and comparable results were obtained.

Tables 6.16 through 6.18 demonstrate that normalized weights that are used to combine hypotheses can indeed be interpreted as the reliability of the classification. Furthermore, as we note from Tables 6.16 through 6.18 for most cases, the reliability levels for correctly classified instances were typically high, whereas those of misclassified instances were generally low or medium, which is undoubtedly comforting to know, when important decisions need to be made.

6.13 Conclusions and Future Work

A new technique, *Learn++*, has been proposed which, in principle, allows any learning algorithm to learn incrementally. Simulations have been performed on a number of databases of varying difficulties to show the feasibility of the approach.

Table 6.18 Learn++ with Mahalanobis confidence analysis on VOC mixture data (II)

#	Ethanol Vote	Octane Vote	Toluene Vote	Xylene Vote	TCE Vote	LEARN++ Decision	Correct Class	Reliability
1	0.981	0.000	0.010	0.008	0.000	ETHANOL	ETHANOL	H
2	1.000	0.000	0.000	0.000	0.000	ETHANOL	ETHANOL	H
3	1.000	0.000	0.000	0.000	0.000	ETHANOL	ETHANOL	H
4	1.000	0.000	0.000	0.000	0.000	ETHANOL	ETHANOL	H
5	1.000	0.000	0.000	0.000	0.000	ETHANOL	ETHANOL	H
6	0.695	0.000	0.305	0.000	0.000	ETHANOL	ETHANOL	M
7	0.997	0.000	0.003	0.000	0.000	ETHANOL	ETHANOL	H
8	0.000	1.000	0.000	0.000	0.000	OCTANE	OCTANE	H
9	0.000	0.910	0.090	0.000	0.000	OCTANE	OCTANE	H
10	0.000	1.000	0.000	0.000	0.000	OCTANE	OCTANE	H
11	0.000	0.769	0.135	0.096	0.000	OCTANE	OCTANE	H
12	0.000	0.527	0.348	0.065	0.061	OCTANE	OCTANE	M
13	0.000	0.824	0.126	0.050	0.000	OCTANE	OCTANE	H
14	0.000	0.914	0.086	0.000	0.000	OCTANE	OCTANE	H
15	0.000	0.000	0.993	0.000	0.007	TOLUENE	TOLUENE	H
16	0.000	0.000	1.000	0.000	0.000	TOLUENE	TOLUENE	H
17	0.000	0.000	0.388	0.384	0.228	TOLUENE	TOLUENE	VL
18	**0.714**	**0.000**	**0.286**	**0.000**	**0.000**	**ETHANOL**	**TOLUENE**	**M**
19	**0.000**	**0.000**	**0.268**	**0.676**	**0.055**	**XYLENE**	**TOLUENE**	**M**
20	**0.000**	**0.000**	**0.303**	**0.697**	**0.000**	**XYLENE**	**TOLUENE**	**M**
21	0.000	0.141	0.594	0.265	0.000	TOLUENE	TOLUENE	M
22	0.000	0.000	1.000	0.000	0.000	TOLUENE	TOLUENE	H
23	0.034	0.000	0.966	0.000	0.000	TOLUENE	TOLUENE	H
24	0.000	0.000	0.111	0.889	0.000	XYLENE	XYLENE	H
25	0.000	0.000	0.080	0.920	0.000	XYLENE	XYLENE	H
26	0.000	0.000	0.250	0.750	0.000	XYLENE	XYLENE	H
27	**0.000**	**0.217**	**0.499**	**0.284**	**0.000**	**TOLUENE**	**XYLENE**	**L**
28	0.000	0.000	0.101	0.899	0.000	XYLENE	XYLENE	H
29	0.040	0.000	0.049	0.911	0.000	XYLENE	XYLENE	H
30	**0.000**	**0.000**	**0.467**	**0.443**	**0.090**	**TOLUENE**	**XYLENE**	**L**
31	0.000	0.000	0.192	0.808	0.000	XYLENE	XYLENE	H
32	0.000	0.000	0.172	0.014	0.815	TCE	TCE	H
33	0.000	0.000	0.330	0.018	0.652	TCE	TCE	M
34	**0.000**	**0.000**	**0.965**	**0.012**	**0.023**	**TOLUENE**	**TCE**	**H**
35	0.000	0.000	0.195	0.013	0.792	TCE	TCE	H
36	0.000	0.000	0.111	0.007	0.882	TCE	TCE	H
37	**0.000**	**0.000**	**0.735**	**0.000**	**0.265**	**TOLUENE**	**TCE**	**M**
38	0.000	0.000	0.083	0.008	0.910	TCE	TCE	H

Learn++ simply attempts to divide a difficult classification problem into smaller sub problems, assigns these sub problems to weak learning algorithms, and combines the outputs using a weighted majority-voting scheme. The underlying assumption is that new data is composed of data from previously unseen or under represented regions of the pattern space, and that simple learning algorithms can be used to learn data coming from these regions.

The main advantage of this approach is that it is very flexible and versatile, and it is independent of the classification algorithm. The algorithm is intuitively simple and easy to implement. It typically runs much faster than strong learning algorithms. Use of weak learning algorithms also eliminates the problem of over fitting since these learners only grossly approximate the instance space.

Indisputably, one of the important features of Learn++ is its ability to predict the reliability of its classification. In Learn++, the built-in voting mechanism is exhibited so that classifications made by winning a strong majority voting are interpreted as high reliability classifications, whereas those winning by narrow margins are interpreted as low reliability classifications.

The main disadvantage of Learn++ is the requirement of significantly high storage capacity. Although it uses weak learning algorithms, which have fewer parameters than their strong counterparts, the total number of parameters can be quite high when an ensemble of these algorithms needs to be saved. This disadvantage, however, is becoming less of an issue, because the current technology allows exponentially increasing data storage capabilities.

Finally, various additional improvements to the algorithm can be proposed as future work. In particular, a more robust distribution update rule and alternate schemes for combining the hypotheses will be topics for future exploration. Learn++ has been diligently tested

using MLP type neural networks since MLP is the most commonly used network architecture in practice. However, evaluating the performance of Learn++ using other classification algorithms also remains to be done.

Another interesting application would be using Learn++ to combine classifiers that are trained with *different* features. Such a scheme would then qualify for not only an incremental learning algorithm, but also a data fusion algorithm. Intuitively, Learn++ would then work as follows: Various databases using different features corresponding to the same classification problem would make the $\mathcal{D}_k$, k=1,...,K databases mentioned in figures 6.3 and 6.6. A set of hypotheses would be generated from each of these databases which could then be combined by a weighted majority voting. However, since databases of different features would be independent of each other, an appropriate distribution update rule and an appropriate weighting scheme for voting would need to be developed. Data fusion using Learn++ constitutes one of the exciting directions for future work.

CHAPTER 7

SUMMARY, CONCLUSIONS AND DISCUSSION

Three major issues have been identified and addressed in this research for the identification of VOCs using piezoelectric crystals as mass sensors. It has been shown that these issues are actually special cases of the more general problems in signal processing, pattern recognition and computational learning. Various approaches to these problems, namely increasing pattern separability for databases of overlapping classes, optimum selection of features and incrementally learning from new data, have been proposed, described and analyzed. The performances of all proposed techniques have been carefully tested on many databases of varying levels of difficulty, including the VOC database.

7.1 Increasing Pattern Separability

In the real world, data are often corrupted with noise and acquired with sensors that are not selective or sensitive enough for the application. This results in a database with overlapping clusters. This phenomenon causes a significant challenge to automated analysis of signals. Developing intelligent algorithms for increasing the intercluster distances within the data is one conceivable solution to this problem. Three such algorithms were presented.

7.1.1 Neuro-Fuzzy Inference Systems

In the first approach a fuzzy inference system was designed and implemented to identify the dominant components of VOC mixtures, followed by a MLP type neural network identifying the secondary components. This approach has the advantage of being very intuitive and

easy to construct, and it is capable of dealing with noisy data. For problems with a reasonable number of input features, the fuzzy inference system can be hand-designed, giving the user an unprecedented amount of control on the automated classification system. However, for complicated tasks with multiple inputs, constructing an intuitive fuzzy inference system becomes increasingly difficult, because such systems require clustering algorithms for determining fuzzy membership functions, effectively removing the control away from the user. The advantage of fuzzy inference systems over neural networks is that they are not *black boxes* like neural networks; hence, their reasoning for a particular classification decision can be traced down. Determining the reason for a particular classification decision is often impossible for all but the most trivial neural networks, due to their massively interconnected structure. However, such a massively interconnected structure of features provides a more powerful classifier then a hand designed fuzzy inference system.

7.1.2 Feature Range Stretching

The second approach, feature range stretching (FRS), was based on the idea of increasing dynamic ranges of the features to increase the separability of patterns by increasing their intercluster distances. The performance of this approach was slightly better than that of the fuzzy system, although FRS also increases intracluster distances, a non-desirable side effect of the algorithm. This approach also has the computational burden of computing a stretching function for each of the features. This can be difficult for signals of high dimensionality.

7.1.3 Nonlinear Cluster Transformations

The third approach, non linear cluster transformation (NCT), was designed to address the shortcomings of the FRS approach. In particular, NCT increases intercluster distances with-

out increasing intracluster distances through translating pattern clusters in the feature space in optimal directions. The optimal directions were first determined by using a training dataset, and then learned using a generalized regression neural network. This method performed better then previous methods. The feasibility of this approach was also demonstrated on various databases.

7.2 Optimal Feature Subset Selection

Methods for increasing pattern separability are useful whenever a database of overlapping clusters needs to be analyzed. Closely related to this issue is the problem of identifying an optimum set of features from a given large pool of features. Selecting the right features can significantly reduce the complexity of the classifier, along with providing computational advantages of dealing with a smaller dataset. Two approaches were proposed for the optimum selection of features. The first approach was using a decision tree algorithm, such as C5.0, based on Quinlan's Iterative Dichotomizer 3 (ID3). In the second approach, an organized search technique, hill climb with wrapper, was proposed to obtain the minimum set of optimal features. The features obtained by this technique were not only smaller in their cardinality, but their classification performance was also better than that of the C5.0 determined features. As in any organized heuristic search technique, hill climbing is computationally complex, and a simple statistical variance analysis was also implemented to reduce this computational burden.

It should be noted that both of these methods, as well as other techniques on feature subset selection, work under the assumption that there is a set of features available and finding the best subset of these features is desired. The real challenge would be identifying the opti-

mum features from a raw dataset, rather than finding the best subset. In other words, the challenge is automatically identifying the right feature extraction scheme for a given dataset and the classification problem. As a simple example, consider the circular regions database, illustrated in Figure 6.7. The features for this database were the *(x,y)* coordinates of individual points. A typical classifier, such as a MLP neural network, takes various weighted combinations of these features and determines nonlinear boundaries as decision surfaces. Another classifier, such as a RBF neural network, would in turn try to generate these decision boundaries through fitting Gaussians to the input/output relationship of the data. In fact, all that is required to correctly cluster these points are their distances from the origin. Therefore, the problem is to develop an algorithm that can analyze the data and identify the distances of points from the origin as useful features.

This challenge, requiring that the algorithm be able to extract features that are optimal for the specific classification problem is of paramount importance for the advancement of automated data analysis. In fact, overcoming the challenge of automatically extracting useful features could effectively make the above described methods for feature subset selection and increasing pattern separability obsolete. It should be noted that this challenge is also of interdisciplinary nature, requiring close collaboration on pattern recognition, signal processing, intelligent agents, and quite possibly neurophysiology research.

7.3 Incremental Learning

Finally, the problem of incrementally learning from new data, without forgetting what has been previously learned, in the absence of the original data has been addressed. This problem arises in many applications where a system is initially trained with a database, and

when a new set of data arrives, the original data is no longer available. Most popular classification algorithms, including MLPs are not capable of learning incrementally, and Learn++ was developed to give all such classification algorithms the capability of learning from new data.

Three different scenarios were considered. In the simplest case, the system was asked to learn from new data that did not include any new class information, simply to improve its classification performance. In the second case, the problem was made considerably more difficult by asking the system to learn patterns coming from a new class not encountered before. In the last case, the restriction of not having the prior data was slightly relaxed, and the algorithm was allowed to keep some statistical information, such as mean and covariance matrix of the data for future training sessions.

Learn++, which is based on the collective performance of an ensemble of classifiers, was tested with a number of databases for each scenario. It was shown that Learn++ was able to learn incrementally from new data regardless of how simple or challenging the database was. Learn++ was also shown to be useful in determining its confidence in its classification decisions.

It is hoped that Learn ++, will provide a valuable tool for all researchers involved in automated classification systems and computational models of learning and take its place among the select few algorithms that are capable of incremental learning.

7.4 Concluding Remarks

As mentioned in the introduction, the research described in this dissertation is an excellent example of interdisciplinary nature of the challenges we face today. In the preceding chapters, the problem of identifying volatile organic compounds has been introduced and it has been shown that it is truly an interdisciplinary problem. It requires expertise in many areas, such as

- analytical chemistry, since the problem involves chemical sensors, their physical and chemical properties,
- electrical engineering, since the problem requires processing of signals and recognizing VOCs from their signature patterns,
- computer science and optimization, since the problem requires expertise on artificial intelligence topics, such as incremental learning and optimum feature selection
- biomedical engineering and olfactory physiology since the problem is closely related to mimicking the human olfactory system.

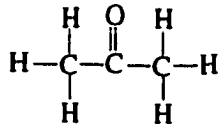
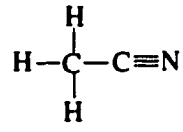
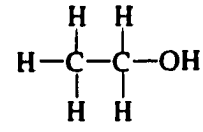
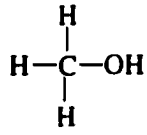
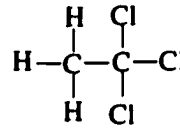
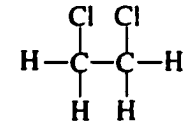
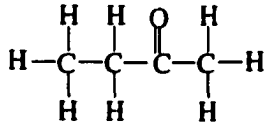
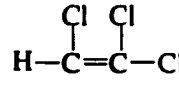
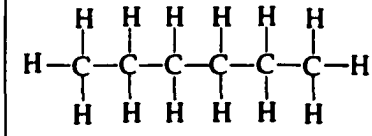
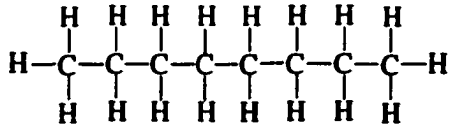
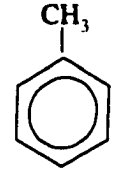
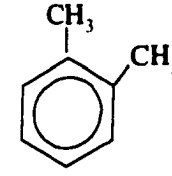
We therefore see that the clear-cut boundaries that once separated various fields, such as life sciences from engineering or computer sciences from physical sciences have been crossed due to the interdisciplinary nature of the problems we face today. We also see new research fields being formed from the merging of various other disciplines, including bioinformatics, genetic engineering, financial engineering, biomedical instrumentation, neural computation and neuro-engineering, among many others.

This merging of disciplines undoubtedly changes the way we conduct research since a strong collaboration is indispensable for the success of the research project. However, it

should also change the way future generations of scientists are trained. Developing new interdepartmental programs and majors such as *electro-biomedical artificial intelligence and neuropysiological computational engineering with emphasis on analytical chemistry* could be a first step.

APPENDIX I

CHEMICAL STRUCTURES OF THE VOCs

Acetone  CH_3CCH_3 (O)	**Acetonitrile**  $CH_3C{\equiv}N$	**Ethanol**  CH_3CH_2OH
Methanol  CH_3OH	**1,1,1-Tricholoroethane**  CH_3CCl_3	**1,2-Dicholoroethane**  $ClCH_2CH_2Cl$
Methylethylketone  $C_2H_5COCH_3$	**Tricholoroethylene**  $ClCH{=}CCl_2$	**Hexane**  C_6H_{14}
Octane  C_8H_{18}	**Toluene**  $C_6H_5CH_3$	**Xylene**  $C_6H_4(CH_3)_2$

APPENDIX II

FNOSE RULEBASE

1. If (PIB is XL) and (DEGA is VS) and (SG is S) and (OV275 is VS) and (PDPP is VS) then (VOC1 is OC)
2. If (APZ is L) and (PIB is XL) and (OV275 is S) and (PDPP is S) then (VOC1 is OC)
3. If (PIB is XL) and (DEGA is VS) and (SG is M) and (OV275 is VS) then (VOC1 is OC)
4. If (DEGA is M) and (SG is M) and (OV275 is M) and (PDPP is M) then (VOC1 is OC)
5. If (DEGA is VS) and (SG is VL) and (OV275 is VS) and (PDPP is VS) then (VOC1 is OC)
6. If (DEGA is VS) and (SG is L) and (OV275 is VS) and (PDPP is VS) then (VOC1 is OC)
7. If (PIB is XL) and (PDPP is VS) then (VOC1 is OC)
8. If (SG is L) and (OV275 is VS) and (PDPP is S) then (VOC1 is OC)
9. If (APZ is XL) and (PIB is M) and (DEGA is L) and (SG is L) and (OV275 is L) and (PDPP is VL) then (VOC1 is XL)
10. If (APZ is VL) and (PIB is M) and (DEGA is L) and (SG is L) and (OV275 is L) and (PDPP is VL) then (VOC1 is XL)
11. If (SG is S) and (OV275 is M) and (PDPP is L) then (VOC1 is XL)
12. If (PIB is L) and (DEGA is L) and (SG is M) and (OV275 is L) and (PDPP is L) then (VOC1 is XL)
13. If (PIB is L) and (DEGA is L) and (SG is M) and (OV275 is M) and (PDPP is L) then (VOC1 is XL)
14. If (APZ is VL) and (PIB is M) and (DEGA is VL) and (SG is VL) then (VOC1 is XL)
15. If (APZ is VL) and (PIB is VL) and (DEGA is M) and (SG is S) and (OV275 is S) then (VOC1 is XL)
16. If (DEGA is XL) and (SG is XL) and (OV275 is XL) then (VOC1 is ET)
17. If (APZ is VS) and (PIB is VS) and (DEGA is XL) and (OV275 is XL) then (VOC1 is ET)
18. If (SG is XL) and (OV275 is VL) and (PDPP is S) then (VOC1 is ET)
19. If (APZ is VS) and (PIB is S) and (DEGA is XL) and (SG is XL) and (OV275 is VL) and (PDPP is XL) then (VOC1 is ET)
20. If (APZ is VS) and (PIB is VS) then (VOC1 is ET)
21. If (PIB is XL) and (SG is XL) and (PDPP is VS) then (VOC1 is ET)
22. If (PIB is S) and (SG is XL) and (PDPP is XL) then (VOC1 is ET)

23. If (APZ is VS) and (PIB is S) and (DEGA is XL) and (SG is XL) and (OV275 is VL) and (PDPP is VL) then (VOC1 is ET)
24. If (DEGA is VL) and (SG is M) and (OV275 is VL) then (VOC1 is TL)
25. If (DEGA is VL) and (SG is L) and (OV275 is VL) then (VOC1 is TL)
26. If (DEGA is XL) and (SG is S) and (OV275 is VL) and (PDPP is XL) then (VOC1 is TL)
27. If (DEGA is VL) and (SG is VL) and (OV275 is VL) then (VOC1 is TL)
28. If (PIB is S) and (DEGA is XL) and (SG is XL) and (OV275 is VL) and (PDPP is VL) then (VOC1 is TL)
29. If (SG is L) and (OV275 is L) and (PDPP is L) then (VOC1 is TL)
30. If (SG is L) and (OV275 is L) and (PDPP is VL) then (VOC1 is TL)
31. If (PIB is M) and (DEGA is L) and (SG is M) and (OV275 is L) and (PDPP is VL) then (VOC1 is TL)
32. If (PIB is M) and (DEGA is VL) and (SG is VL) and (OV275 is L) then (VOC1 is TL)
33. If (PIB is S) and (DEGA is VL) and (SG is XL) and (OV275 is VL) and (PDPP is XL) then (VOC1 is TL)
34. If (DEGA is S) and (SG is S) and (OV275 is S) then (VOC1 is TCE)
35. If (APZ is L) and (PIB is L) and (DEGA is M) and (SG is M) and (OV275 is M) and (PDPP is L) then (VOC1 is TCE)
36. If (APZ is M) and (PIB is L) and (DEGA is L) and (SG is M) and (OV275 is L) then (VOC1 is TCE)
37. If (PIB is VL) and (DEGA is M) and (SG is S) and (PDPP is M) then (VOC1 is TCE)
38. If (DEGA is S) and (SG is M) and (OV275 is S) and (PDPP is M) then (VOC1 is TCE)
39. If (SG is S) and (OV275 is VS) and (PDPP is M) then (VOC1 is TCE)
40. If (PIB is VL) and (DEGA is M) and (SG is L) then (VOC1 is TCE)
41. If (APZ is L) and (PIB is VL) and (PDPP is M) then (VOC1 is TCE)
42. If (APZ is XL) and (PIB is VL) and (PDPP is M) then (VOC1 is TCE)
43. If (DEGA is S) and (PDPP is S) then (VOC1 is TCE)
44. If (APZ is VL) and (PIB is VL) and (PDPP is M) then (VOC1 is TCE)
45. If (PIB is L) and (DEGA is L) and (SG is VL) and (PDPP is L) then (VOC1 is TCE)
46. If (PIB is M) and (SG is XL) and (OV275 is VL) then (VOC1 is TCE)
47. If (DEGA is VS) and (SG is S) and (OV275 is VS) and (PDPP is S) then (VOC1 is TCE)
48. If (APZ is VL) and (PIB is XL) and (DEGA is VS) and (SG is VS) and (OV275 is VS) and (PDPP is S) then (VOC1 is TCE)

49. If (APZ is M) and (PIB is VL) and (DEGA is M) then (VOC1 is TCE)
50. If (APZ is VL) and (PIB is XL) and (DEGA is VS) and (SG is M) and (OV275 is VS) and (PDPP is S) then (VOC1 is TCE)
51. If (APZ is M) and (PIB is VL) and (DEGA is M) and (SG is M) and (OV275 is M) and (PDPP is L) then (VOC1 is TCE)
52. If (APZ is VL) and (PIB is VL) and (DEGA is S) and (SG is L) and (OV275 is S) and (PDPP is M) then (VOC1 is OC)
53. If (APZ is VL) and (PIB is XL) and (DEGA is VS) and (SG is VL) and (OV275 is L) and (PDPP is S) then (VOC1 is OC)
54. If (APZ is VL) and (PIB is VL) and (DEGA is M) and (SG is S) and (OV275 is VL) and (PDPP is S) then (VOC1 is OC)
55. If (APZ is XL) and (PIB is M) and (DEGA is VL) and (SG is VL) and (OV275 is VL) and (PDPP is VL) then (VOC1 is XL)

APPENDIX III

FUZZY MEMBERSHIP FUNCTIONS

OCTANE

OCTANE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
TOLUENE	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	S	VL	L	L	L	VL	L	L	L	L	L	L	L	VL	L	L
PIB	XL	XL	XL	XL	XL	XL	XL	XL	VL	XL	XL	XL	XL	XL	XL	XL
DEGA	VS	VS	VS	VS	S	VS	VS	VS	M	S	VS	VS	M	S	VS	VS
SG	S	S	S	S	L	M	S	S	M	M	M	S	M	M	M	M
OV275	VS	VS	VS	VS	S	VS	VS	VS	M	S	VS	VS	M	S	S	VS
PDPP	S	VS	VS	VS	S	S	VS	VS	M	S	S	S	M	M	S	S

OCTANE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
ETHANOL	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	VL	XL	VL	VL	VL	VL	VL	VL	XL	VL	VL	VL	VL	VL	VL	L
PIB	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL
DEGA	VS	VS	VS	VL	VS	VS	VS	VS	S	VS	VS	VS	M	VS	VS	VS
SG	VL	S	S	S	VL	L	M	M	XL	VL	L	M	XL	VL	L	L
OV275	VS	VS	VS	VS	VS	VS	VS	VS	S	VS	VS	VS	L	S	VS	VS
PDPP	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS

OCTANE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
TCA	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	VL	VL	L	L	XL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL
PIB	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	VL	XL	XL	XL
DEGA	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	S	VS	VS	VS
SG	M	S	S	S	L	M	S	S	L	L	M	M	L	L	M	M
OV275	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	S	VS	VS	VS
PDPP	VS	VS	VS	VS	S	VS	VS	VS	S	S	VS	VS	M	S	S	VS

OCTANE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
MEK	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	VL	VL	VL	L	XL	VL	VL	L	VL	VL	L	VL	VL	L	L	L
PIB	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	VL	XL	XL	XL
DEGA	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	S	VS	VS	VS
SG	M	S	S	S	VL	M	M	M	VL	VL	M	M	XL	VL	L	L
OV275	VS	VS	VS	VS	S	VS	VS	VS	L	S	VS	VS	VL	S	VS	VS
PDPP	VS	VS	VS	VS	VS	VS	VS	VS	S	VS	VS	VS	S	S	VS	VS

OCTANE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
ACN	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	VL	VL	L	L	VL	VL	L	L	VL	VL	L	L	VL	VL	L	L
PIB	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	VL	XL	XL	XL
DEGA	VS	VS	VS	VS	VS	VS	VS	VS	S	VS	VS	VS	M	VS	VS	VS
SG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
OV275	VS	VS	VS	VS	S	VS	VS	VS	L	S	VS	VS	VL	S	S	VS
PDPP	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	S	VS	VS	VS

Figure A3. 1 Octane fuzzy membership functions

XYLENE

XYLENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
MEK	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	VL	VL	L	XL	VL	VL	L	XL	VL	VL	L	XL	VL	L	L
PIB	M	L	L	L	M	L	L	L	M	M	L	L	M	M	L	L
DEGA	L	L	L	L	L	L	L	L	VL	L	L	L	VL	L	L	L
SG	L	S	S	S	L	M	M	S	VL	L	M	M	VL	L	M	M
OV275	L	L	M	M	L	L	M	M	VL	L	L	L	VL	L	L	L
PDPP	VL	L	L	L	VL	L	L	L	VL	L	L	L	VL	VL	L	L

XYLENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
TCA	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	VL	L	L	L	VL	L	L	L	VL	L	L	L	VL	L	L	L
PIB	L	VL	VL	VL	L	L	VL	VL	L	L	L	L	M	L	L	L
DEGA	M	L	L	M	L	L	L	M	L	L	L	M	L	L	L	L
SG	S	S	S	S	M	M	S	S	M	M	M	S	L	M	M	M
OV275	M	M	M	M	M	M	M	M	M	M	M	M	L	M	M	M
PDPP	L	L	L	L	L	L	L	L	VL	L	L	L	VL	L	L	L

XYLENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
ACN	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	VL	L	L	XL	VL	L	L	VL	L	L	L	VL	L	L	L
PIB	L	VL	VL	VL	L	VL	VL	VL	L	VL	VL	VL	VL	VL	VL	VL
DEGA	M	M	M	M	M	M	M	M	L	M	M	M	M	M	M	M
SG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
OV275	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
PDPP	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L

XYLENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
ETHANOL	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	VL	VL	L	L	XL	L	L	L	VL	L	L	L	VL	L	L	L
PIB	L	VL	VL	VL	L	L	L	VL	M	L	L	L	M	L	L	L
DEGA	L	L	M	M	L	L	L	M	VL	L	L	L	VL	L	L	L
SG	M	S	S	S	M	M	S	S	VL	M	M	S	VL	L	M	S
OV275	L	M	M	M	L	M	M	M	L	L	M	M	VL	L	L	M
PDPP	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L

XYLENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
HEXANE	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	VL	L	L	L	VL	VL	L	L	VL	VL	L	L	VL	VL	L	L
PIB	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL
DEGA	M	L	M	M	M	M	M	M	M	M	M	M	M	M	M	M
SG	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
OV275	M	M	M	M	M	M	M	M	S	M	M	M	S	S	M	M
PDPP	L	L	L	L	L	L	L	L	M	L	L	L	M	L	L	L

Figure A3. 2 Xylene fuzzy membership functions

ETHANOL

ETHANOL	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
ACN	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS
PIB	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS
DEGA	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL
SG	XL	XL	XL	XL	S	VL	XL	XL	VS	S	VL	XL	VS	VS	M	VL
OV275	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL
PDPP	M	M	S	S	M	M	M	S	L	M	M	M	L	M	M	M

ETHANOL	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
HEXANE	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	S	VS	VS	VL	L	VS	VS	VL	L	M	S	L	VL	S	M
PIB	S	VS	VS	VS	VL	M	S	VS	XL	VL	M	S	XL	XL	VL	M
DEGA	XL	XL	XL	XL	L	XL	XL	XL	M	VL	XL	XL	S	L	VL	XL
SG	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL
OV275	VL	XL	XL	XL	VL	VL	VL	XL	M	VL	VL	VL	S	L	VL	VL
PDPP	S	S	S	S	S	S	S	S	VS	S	S	S	VS	VS	S	S

ETHANOL	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
TCA	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	M	VS	VS	VS	S	VS	VS	VS	VS	VS	VS	VS	S	VS	VS	VS
PIB	S	VS	VS	VS	S	S	VS	VS	S	S	S	S	S	S	S	S
DEGA	XL	XL	XL	XL	XL	XL	XL	XL	VL	XL	XL	XL	VL	VL	XL	XL
SG	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL
OV275	VL	VL	XL	XL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL
PDPP	XL	VL	VL	L	XL	XL	VL	XL	XL	XL	XL	VL	XL	XL	XL	XL

ETHANOL	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
MEK	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS
PIB	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS
DEGA	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL
SG	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL
OV275	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL
PDPP	VL	L	M	M	XL	VL	L	L	XL	XL	VL	L	XL	XL	VL	VL

Figure A3. 3 Ethanol fuzzy membership functions

TOLUENE

TOLUENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
ACN	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	VL	L	M	XL	VL	M	M	M	M	M	S	S	M	S	S
PIB	S	S	M	M	S	S	S	M	VS	S	S	S	VS	S	S	S
DEGA	VL	VL	VL	VL	XL	VL	VL	VL	XL	VL	VL	VL	XL	XL	VL	VL
SG	M	M	M	L	M	M	M	M	S	M	M	M	VS	S	M	M
OV275	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL
PDPP	VL	VL	VL	VL	XL	XL	VL	VL	XL	XL	XL	XL	XL	XL	XL	XL

TOLUENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
ETHANOL	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	VL	L	L	XL	VL	L	M	XL	VL	L	M	VL	L	M	M
PIB	S	S	M	M	S	S	M	M	S	S	S	M	S	S	S	S
DEGA	VL	VL	VL	VL	XL	VL	VL	VL	XL	VL	VL	VL	XL	XL	VL	VL
SG	VL	VL	L	L	XL	VL	VL	L	XL	VL	VL	VL	XL	XL	VL	VL
OV275	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL
PDPP	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL

TOLUENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
HEXANE	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	VL	L	M	XL	VL	L	M	XL	VL	L	M	VL	L	M	M
PIB	M	M	M	M	L	M	M	M	VL	L	L	M	VL	VL	L	L
DEGA	VL	VL	L	L	L	L	L	L	M	L	L	L	M	M	L	L
SG	L	L	L	L	L	L	M	M	L	L	L	M	L	L	L	L
OV275	L	L	L	L	L	L	L	L	M	L	L	L	M	L	L	L
PDPP	L	VL	VL	VL	L	L	VL	VL	M	L	L	VL	M	L	L	L

TOLUENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
TCA	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	L	L	M	S	L	M	M	L	M	M	M	L	M	M	S
PIB	S	M	M	M	S	M	M	M	M	M	M	M	M	M	M	M
DEGA	VL	VL	VL	VL	XL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL
SG	L	L	L	L	M	L	L	VL	VL	VL	L	L	VL	VL	L	L
OV275	VL	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
PDPP	XL	VL	VL	VL	L	VL	VL	VL	XL	XL	VL	VL	XL	XL	VL	VL

TOLUENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
MEK	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	VL	M	S	VS	M	S	S	VS	S	S	S	VS	S	S	S	S
PIB	S	S	S	VS	S	S	S	S	M	S	S	S	M	M	S	S
DEGA	VL	VL	XL	XL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL
SG	VL	XL	XL	XL	VL	VL	VL	XL	VL	VL	VL	VL	L	VL	VL	VL
OV275	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL	VL
PDPP	XL	XL	XL	XL	XL	XL	XL	XL	VL	XL	XL	XL	VL	XL	XL	XL

Figure A3. 4 Toluene fuzzy membership functions

TOLUENE

TOLUENE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
TCE	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	VL	VL	VL	VL	L	L	L	L	L	L	L	M	M	M	M
PIB	VL	VL	VL	XL	L	VL	VL	VL	L	L	VL	VL	L	L	L	VL
DEGA	M	S	S	S	L	M	M	S	L	M	M	M	L	L	M	M
SG	S	S	S	S	S	S	S	S	M	M	S	S	M	M	S	M
OV275	M	S	S	S	M	M	S	S	L	M	M	M	L	L	M	M
PDPP	M	M	M	M	L	L	M	M	L	L	L	L	VL	L	L	L

TCE

TCE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
TCA	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	VL	VL	L	VL	VL	L	L	L	VL	L	L	L	L	L	M
PIB	VL	XL	XL	XL	VL	VL	VL	XL	VL	VL	VL	VL	L	VL	VL	VL
DEGA	S	S	S	VS	M	S	S	S	M	M	S	S	L	M	M	S
SG	M	S	S	S	L	M	S	S	L	L	M	M	VL	L	L	M
OV275	S	VS	VS	VS	S	S	VS	VS	S	S	S	VS	M	S	S	S
PDPP	M	M	M	M	M	M	M	M	L	L	M	M	L	L	L	M

TCE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
TCA	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	VL	VL	L	VL	VL	L	L	L	L	M	L	S	M	M	M
PIB	VL	XL	XL	XL	VL	VL	VL	XL	M	VL	VL	VL	M	L	VL	VL
DEGA	S	S	S	VS	M	S	S	S	L	M	M	S	VL	L	M	S
SG	L	M	S	S	VL	L	M	M	XL	VL	L	M	XL	VL	VL	M
OV275	M	S	S	VS	L	M	S	S	VL	L	L	M	VL	VL	L	L
PDPP	M	M	M	M	M	M	M	M	L	M	M	M	VL	L	L	M

TCE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
ETHANOL	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	XL	VL	VL	XL	XL	VL	VL	XL	VL	L	L	VL	VL	VL	L
PIB	VL	XL	XL	XL	VL	VL	XL	XL	L	VL	XL	XL	M	VL	VL	VL
DEGA	S	S	S	S	L	S	S	S	VL	L	M	S	VL	L	M	M
SG	M	S	S	S	VL	L	M	S	XL	VL	L	M	XL	VL	VL	L
OV275	S	VS	VS	VS	L	S	S	S	VL	L	M	S	VL	L	L	M
PDPP	S	S	S	M	M	M	M	M	M	M	M	M	M	M	M	M

TCE	150	300	500	700	150	300	500	700	150	300	500	700	150	300	500	700
HEXANE	150	150	150	150	300	300	300	300	500	500	500	500	700	700	700	700
APZ	XL	VS	VL	L	XL	VL	VL	L	VL	VL	VL	L	VL	VL	VL	L
PIB	XL	VS	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL	XL
DEGA	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS
SG	S	VS	VS	S	S	S	S	S	S	S	S	S	M	S	S	S
OV275	VS	XL	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS	VS
PDPP	S	VS	S	S	S	S	S	S	S	S	S	S	S	S	S	S

Figure A3. 5 TCE fuzzy membership functions

APPENDIX IV

LEARN++ C-SCAN CLASSIFICATION OF UWI SIGNALS

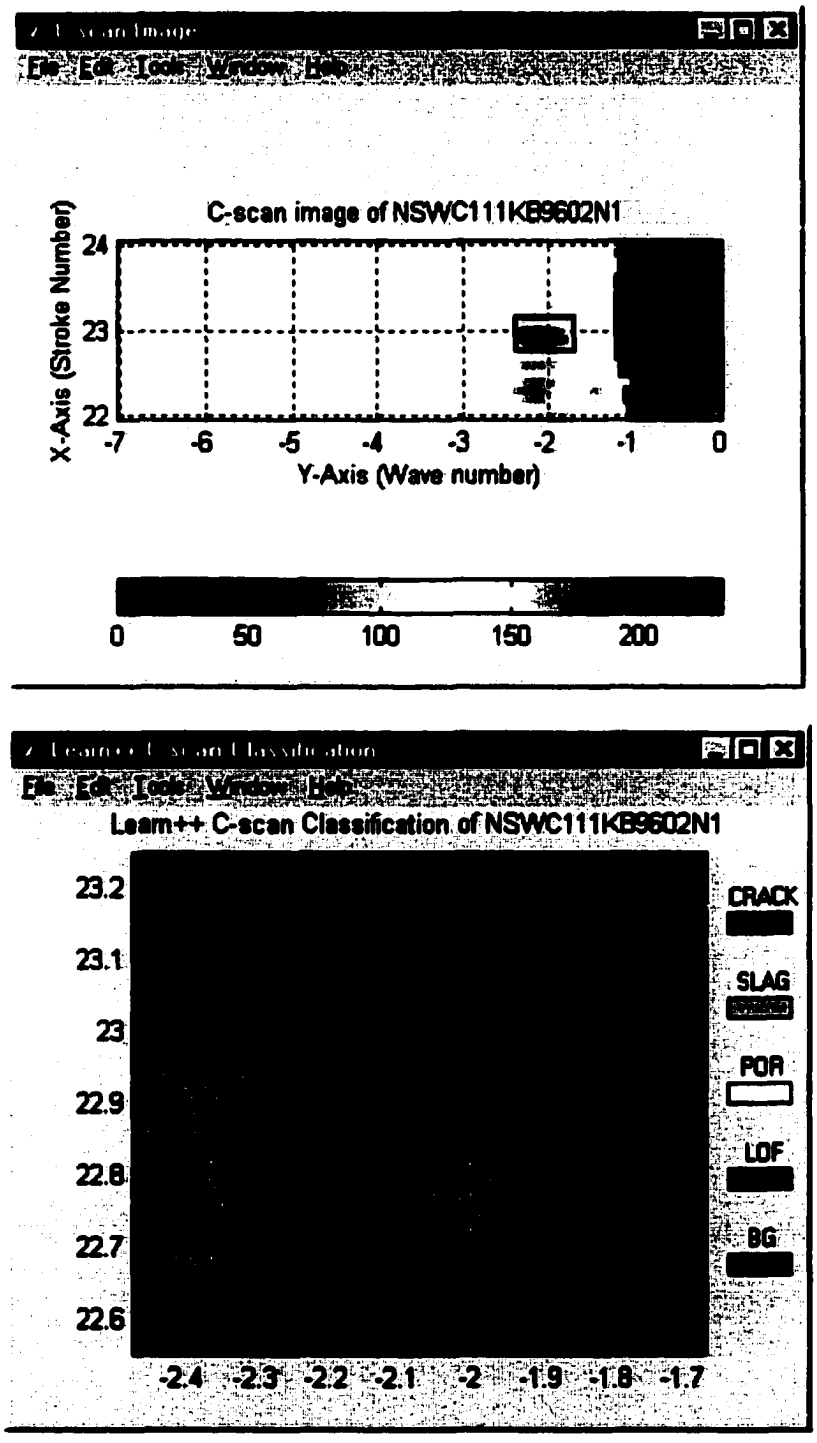


Figure A4. 1 Original C-scan and Learn++ classification, correct class: Crack

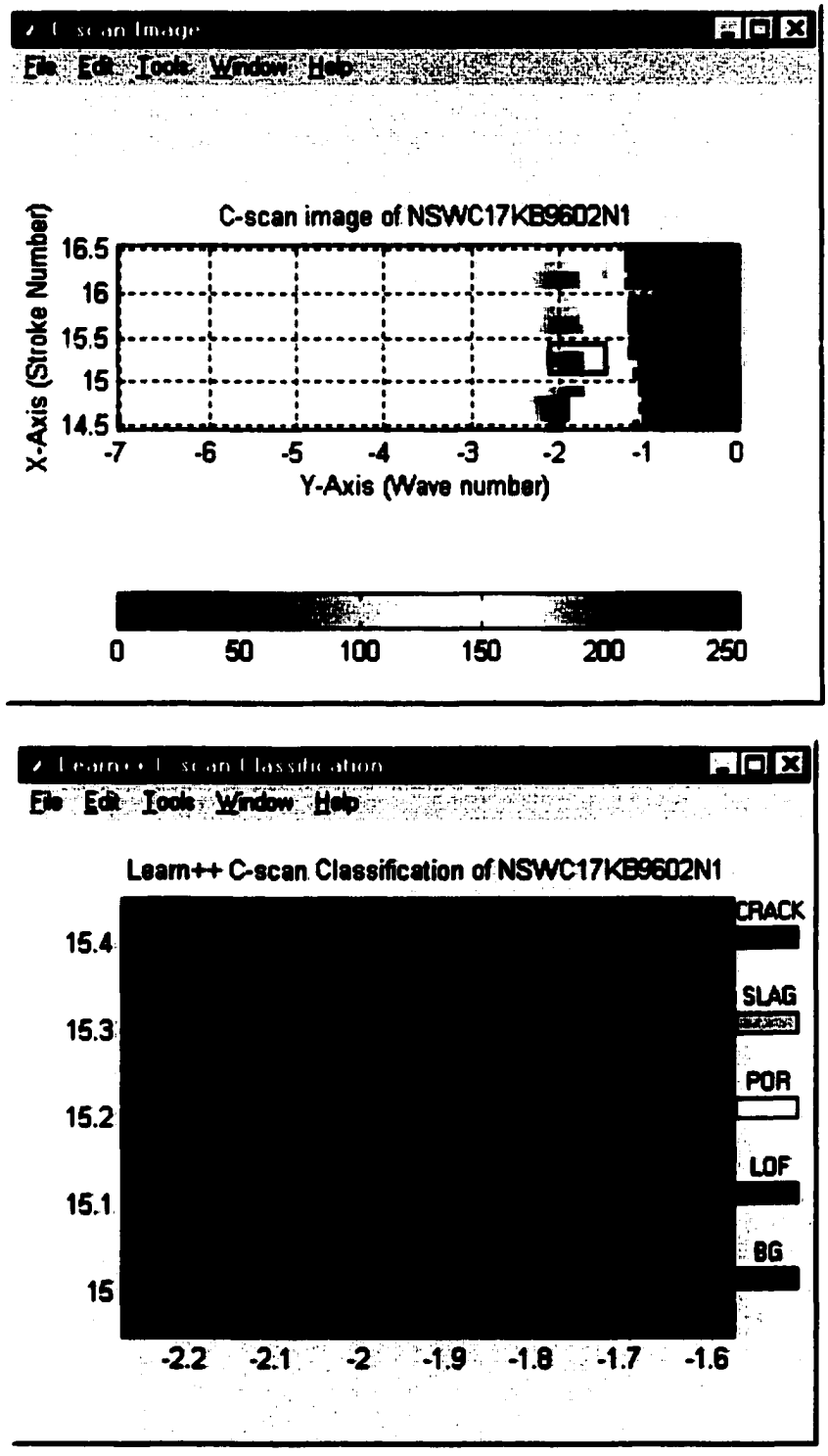
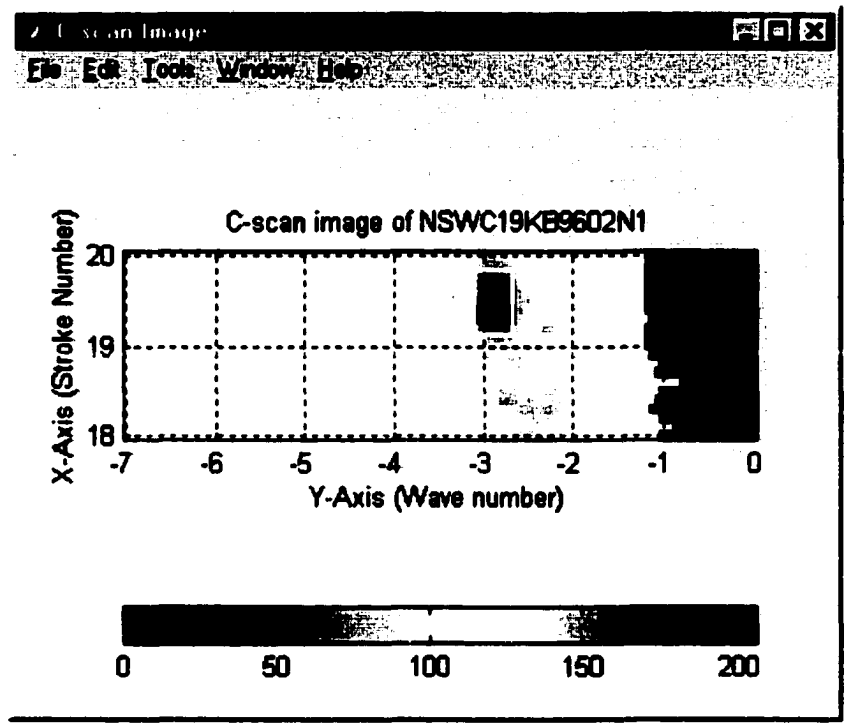


Figure A4. 2 Original C-scan and Learn++ classification, correct class: Crack



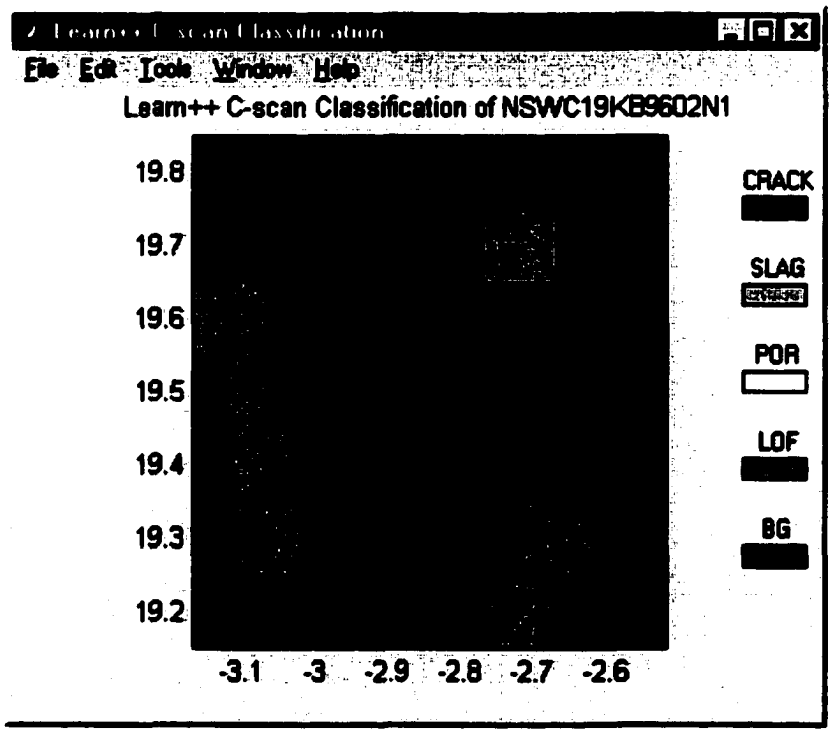


Figure A4. 3 Original C-scan and Learn++ classification, correct class: Crack

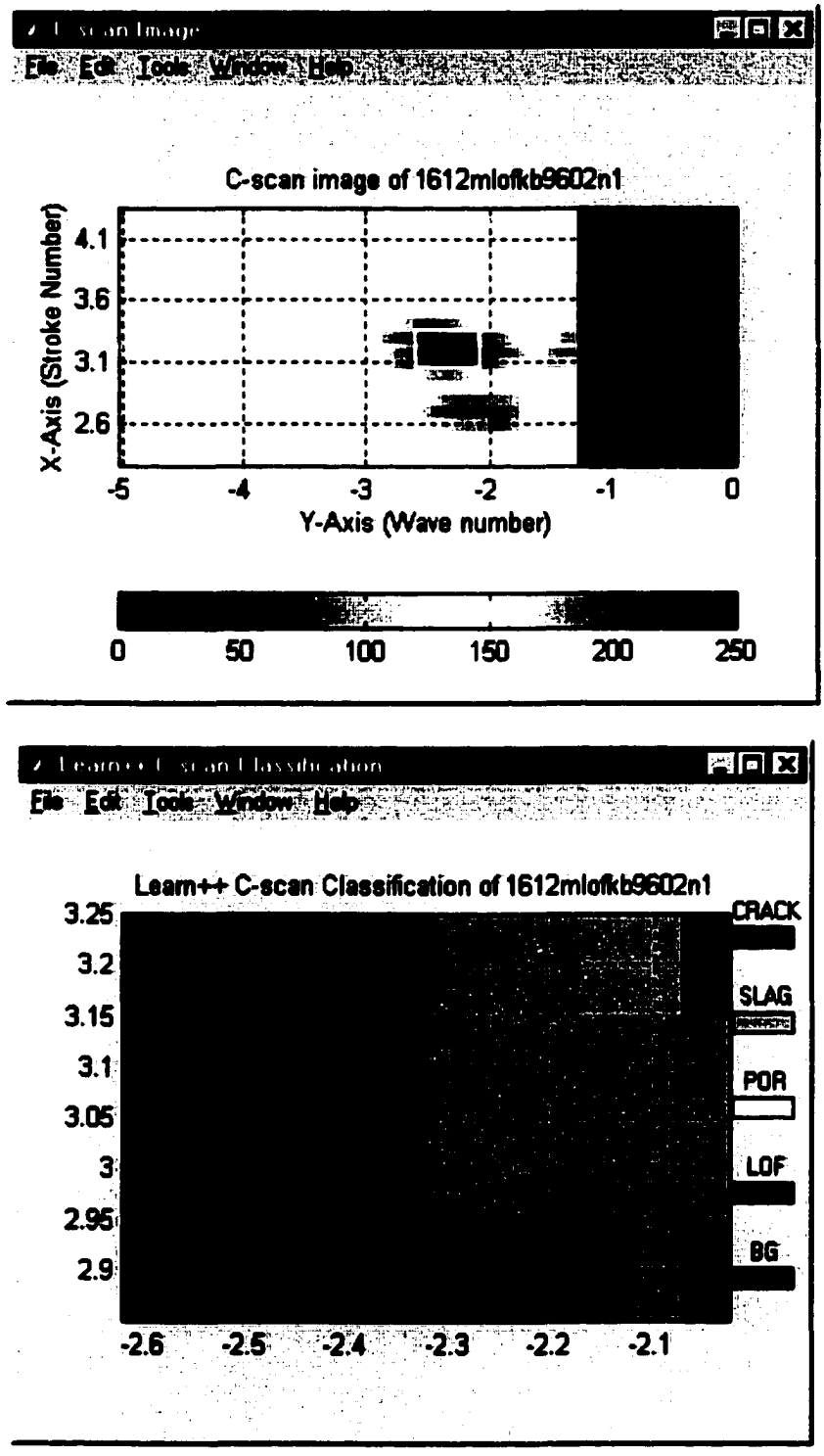


Figure A4. 4 Original C-scan and Learn++ classification, correct class: LOF

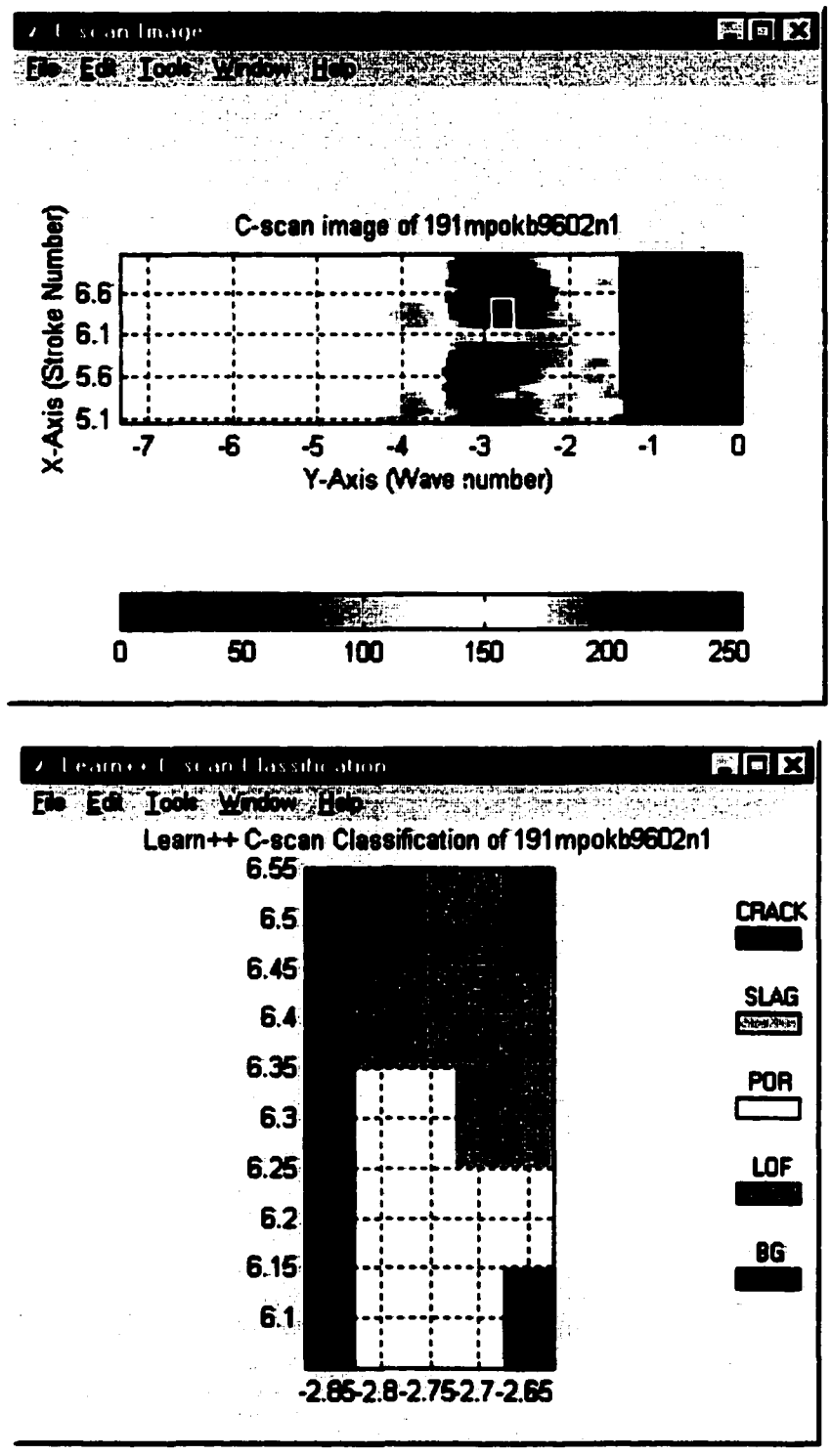


Figure A4. 5 Original C-scan and Learn++ classification, correct class: Porosity

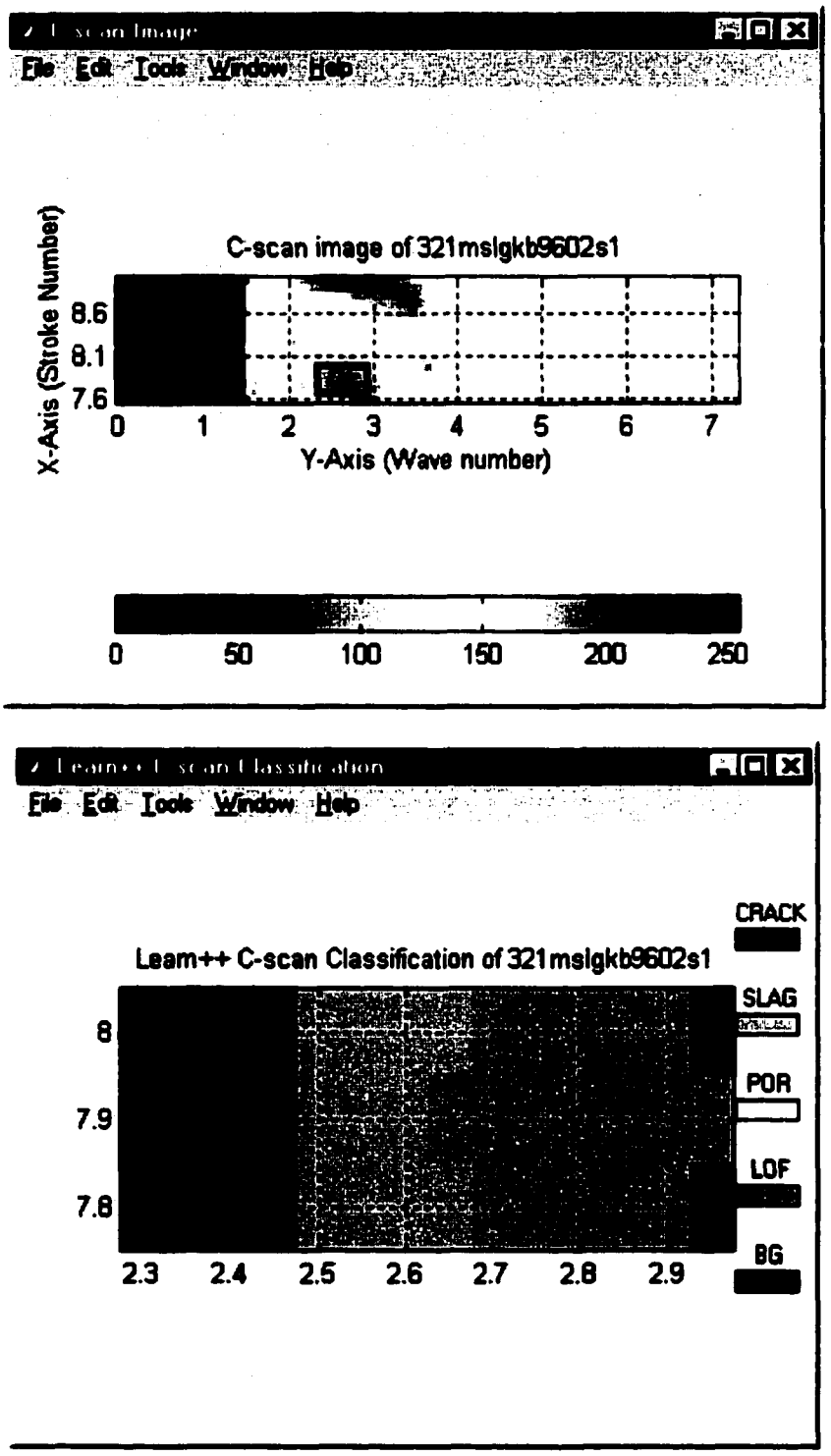


Figure A4. 6 Original C-scan and Learn++ classification, correct class: Slag

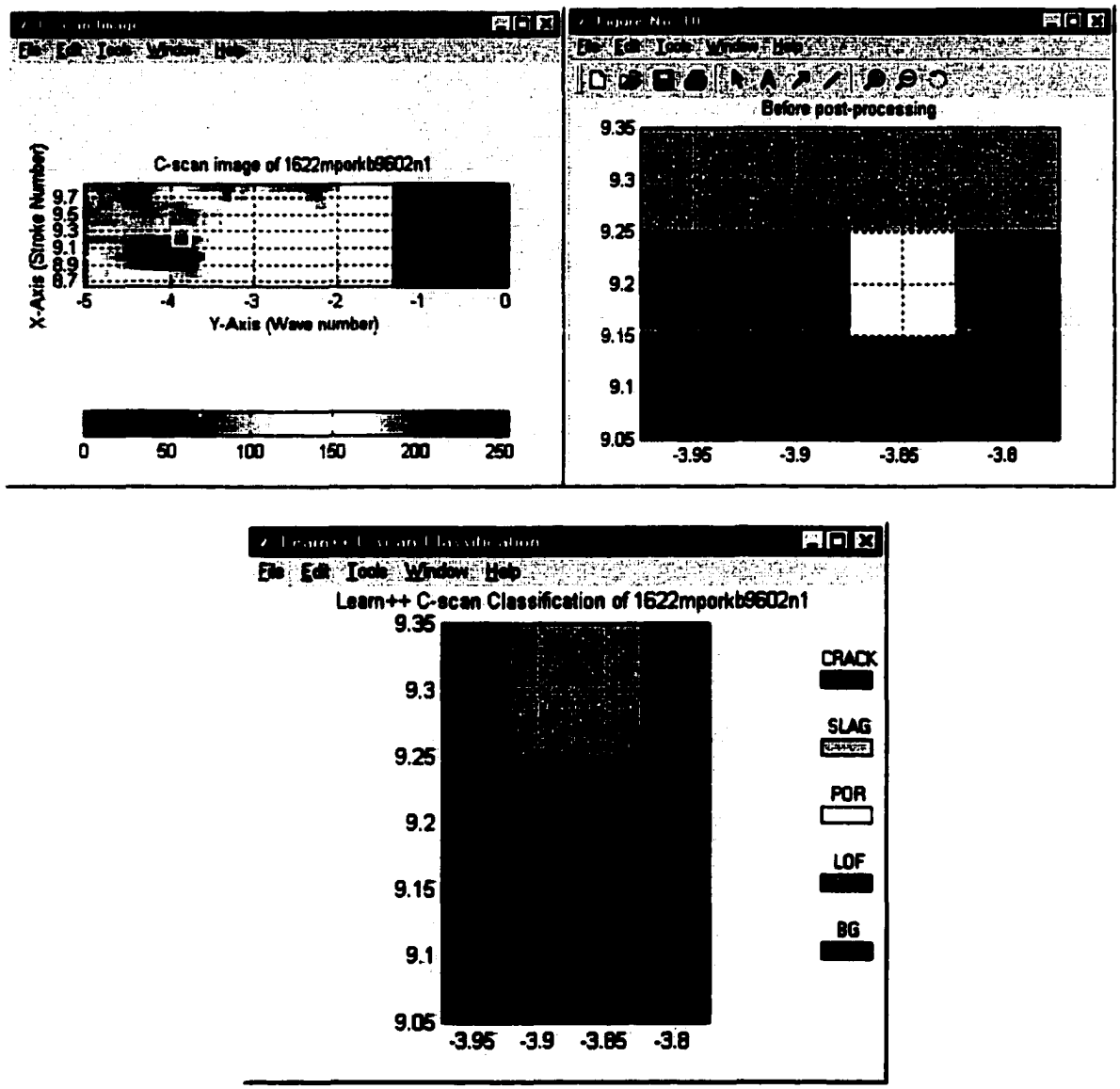


Figure A4. 7 Original C-scan and Learn++ classification, correct class: Porosity

The Learn++ classification before post processing is also provided for this sample. The porosity indication for this sample was known to be extremely small (about 0.1 inches long, inside the white rectangular area in C-scan), and Learn++ pinpointed the correct location of the porosity indication. However, post processing (modified median filtering) wiped out the porosity indication, since it is completely surrounded by crack and LOF indications.

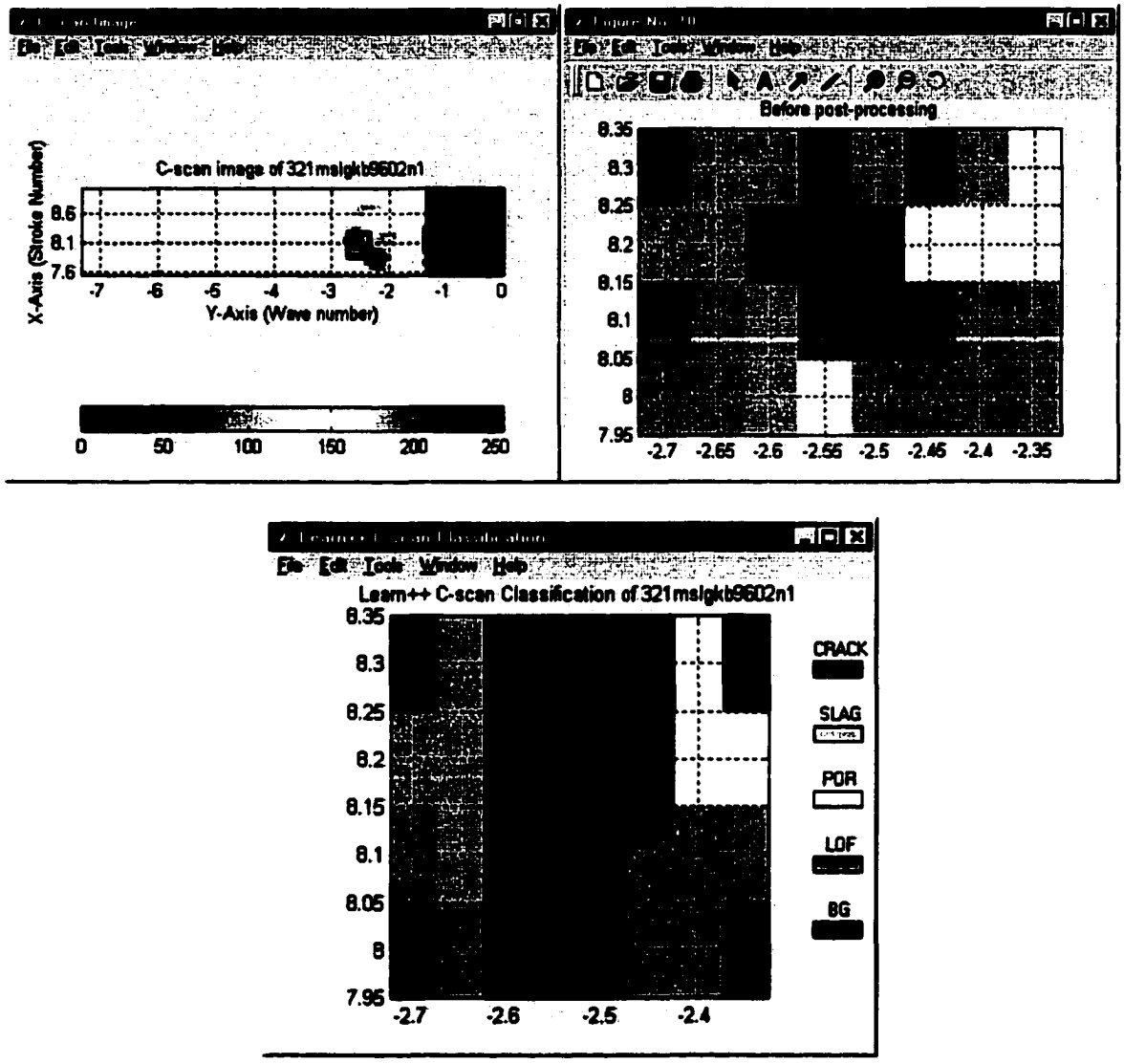


Figure A4. 8 Original C-scan and Learn++ classification, correct class: Slag

This is an example of unknown classification, since all flaw types have been found by Learn++ in the indicated region of interest shown by the black box.

REFERENCES

[1] C. Di Natale, F.A.M, Davide, A. D'Amico, P. Nelli, S. Groppelli, and G. Sberveglieri, "An electronic nose for the detection of the vineyard of a red wine,", *Sensors and Actuators, B* vol. 33, pp. 83-88, 1996.

[2] C. Di Natale, A. Macagnano, F. Davide, A. D'Amico, R. Paolesse, T. Boschi, M. Faccio, and G. Ferri, "An electronic nose for food analysis," *Sensors and Actuators, B* vol. 44, pp. 521-526, 1997.

[3] P.M. Schweizer-Berberich, S. Vaihinger, and W. Gopel, "Characterization of food freshness with sensor arrays," *Sensors and Actuators, B* vol. 18-19, pp. 282-290, 1994.

[4] S. Ashley, "Searching for land mines," *Mechanical Engineering* vol. 114, no.4, pp. 62-67, 1996.

[5] J.R. Stetter, P.C. Jurs, and S.L. Rose, "Detection of hazardous gases and vapors: Pattern recognition analysis of data from an electrochemical sensor array," *Analytical Chemistry*, vol. 58, pp. 860-866, 1986.

[6] J.W. Grate, S. Rose-Pehrsson, D.L. Venezky, M. Klutsy, and H. Wohltjen, "Smart sensor system for trace organophosphorus and organosulfur vapor detection employing a temperature controlled array of surface acoustic wave sensors, automated sample preconcentration, and pattern recognition," *Analytical Chemistry*, vol. 65, pp.1868-1881, 1993.

[7] S.L. Rose-Pehrsson, J.W. Grate, D.S. Ballantine, P.C. Jurs, "Detection of hazardous vapors including mixtures using pattern recognition analysis of responses from surface acoustic wave devices," *Analytical Chemistry*, vol. 60, pp. 2801- 2811, 1988.

[8] C. A. Rowe-Taitt, J.P.Golden, M.J. Feldstein, J.J. Cras, K.E. Hoffman, F.S. Ligler, "Array biosensor for detection of biohazards," *Biosensors and Bioelectronics*, vol. 14, pp. 785-794, 2000.

[9] H. Nanto, Y. Douguchi, K. Yokoi, T. Mukai, J. Fujioka, E. Kusano, A. Kinbara, "Smart electronic nose using polymer-film coated quartz resonator gas sensor for identification of harmful gases," *Proceedings of SPIE, Internal Standardization and Calibration Architectures for Chemical Sensors*, vol. 3856, pp. 317-327, 1999.

[10] V. Grimaldi and J.L. Politano, " Illicit material detector based on gas sensors and neural networks," *Proceedings of SPIE, Chemistry and Biology Based Technologies for Contraband Detection*, vol. 2937, pp. 90-99, 1997.

[11] K. Bodenhöfer, A. Hierlemann, R. Schlunk, and W. Göpel, "New method of vaporizing volatile organics for gas tests," *Sensors and Actuators B*, vol. 45, pp. 259-264, 1997.

[12] M. Holmberg, F.A.M. Davide, C. DiNatale, A. D'Amico, F. Winquist, and I. Lundström, "Drift counteraction in odour recognition applications: lifelong calibration method," *Sensors and Actuators B*, vol. 42, pp. 185-194, 1997.

[13] D. M. Wilson and S.P. DeWeerth, "Signal processing for improving gas sensor response time," *Sensors and Actuators B*, vol. 41, pp. 63-70, 1997.

[14] R. Gutierez-Osuna, H. Troy Nagle, and S.S. Schiffman, "Trnasient response analysis of an electronic nose using multi-exponential models," *Sensors and Actuators B*, vol. 61, pp. 170-182, 1999.

[15] L. Ratton, T. Kunt, T. McAvoy, T. Fuja, R. Cavicchi, and S. Semancik, "A comparative study of signal processing techniques for clustering microsensor data (a first step towards an artificial nose)," *Sensors and Actuators B*, vol. 41, pp. 105-120, 1997.

[16] K. Nakamura, T. Nakamoto, and T. Moriizumi, " Prediction of quartz crystal microbalance gas sensor responses using a computational chemistry method," *Sensors and Actuators B*, vol. 61, pp. 6-11, 1999.

[17] M.E.H Amrani, R. M., Dowdeswell, P.A. Payne, and K.C. Persaud, " An intelligent gas sensing system," *Sensors and Actuators B*, vol. 44, pp. 512-516, 1997.

[18] J. W. Gardner, "Detection of Vapours and Odours from a multisensor array using pattern recognition: Part1. Principle component and cluster analysis," *Sensors and Actuators B*, vol. 4, pp. 109-115, 1991.

[19] J. Auge, P. Hauptmann, J. Hartmann, S. Rösler, and R. Lucklum, "Versatile microcontrolled gas sensor array system using the quartz microbalance principle and pattern recognition methods," *Sensors and Actuators B*, vol. 26-27, pp. 181-186, 1995.

[20] K.C. Persaud, S. M. Khaffaf, J. S. Payne, A.M. Pisanelli, D.H. Lee, and H.G. Byun," Sensor array techniques for mimicking the mammalian olfactory system," *Sensors and Actuators B*, vol. 35-36, pp. 267-273, 1996.

[21] G. Nieblimg, " Identification of gases with classical pattern recognition methods and artificial neural networks," *Sensors and Actuators B*, vol. 18-19, pp. 259-263, 1994.

[22] E.T. Zellars, S.A. Batterman, M. Han, and S.J. Patrash, "Optimal coating selection for the analysis of organic vapor mixtures with polymer coated surface acoustic wave sensor arrays," *Analytical Chemistry*, vol. 67, pp. 1092-1106, 1995.

[23] G. Barkó, B. Papp, and J. Hlavay, " Application of pattern recognition and piezoelectric sesnsor array for the detection of organic compounds," *Talanta*, vol. 42, no. 3, pp. 475-482, 1995.

[24] J.W. Gardner and P.N. Bartlett, "A brief history of electronic noses," *Sensors and Actuators B*, vol.18-19, pp. 211-220, 1994.

[25] N. Ryman-Tubb, " They all stink!: Chemometrics and the neural approach," *Proceeding of SPIE (Virtual Intelligence)*, vol. 2878, pp. 117-127, 1996.

[26] D.D. Vlachos, D.K. Fragoulis, and J.N. Avaritsiotis, "An adaptive neural network topology for degradation compensation of thin film tin oxide gas sensors," *Sensors and Actuators B*, vol. 45, pp. 223-228, 1997.

[27] Z. Wang, J. Hwang, and B.R. Kowalski, "ChemNets: Theory and application," *Analytical Chemistry*, vol. 67, pp. 1497-1504, 1995.

[28] H. Hong, H.W. Shin, H.S. Park, D.H. Yun, C.H. Kwon, K. Lee, S. Kim, and T. Moriizumi, "Gas identification using micro gas sensor array and neural network pattern recognition," *Sensors and Actuators B*, vol. 33, pp. 68-71, 1996.

[29] B. Yea, T. Osaki, K. Sugahara, and R. Konishi, "The concentration estimation of inflammable gases with a semiconductor gas sensor utilizing neural networks and fuzzy inference," *Sensors and Actuators B*, vol. 41, pp. 121-129, 1997.

[30] H.S. Abdel-Aty-Zohdy and M. Al-Nsour, " Reinforcement learning neural network circuits for electronic nose," *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS99)*, vol. 5, pp. 379-382, 1999.

[31] M. Nakamura and I. Sugimoto, "A neural network model for an electronic nose based on quartz crystal microbalance sensors," *Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN99)*, vol. 2, pp. 649-654, 1999.

[32] D. Vlachos and J. Avaritsiotis, "Fuzzy neural networks for gas sensing," *Sensors and Actuators B*, vol. 33, pp. 77-82, 1996.

[33] W. Ping and X. Jun, " A novel recognition method for electronic nose using artificial neural network and fuzzy recognition," *Sensors and Actuators B*, vol. 37, pp. 169-174, 1996.

[34] E. Llobet, E.L. Hines, J.W. Gardner, P.N. Bartlett, and T.T.Mottram, "Fuzzy ARTMAP based electronic nose data analysis," *Sensors and Actuators B*, vol. 61, pp. 183-190, 1999.

[35] D. Dumitrescu, B. Lazzerini, F. Marcelloni, " Fuzzy hierarchical approach to odor classification," *Proceedings of SPIE, Ninth Workshop on Virtual Intellgence / Dynamic Neural Networks*, vol. 3728, pp. 384-395, 1999.

[36] B. G. Kermani, "Using neural networks and genetic algorithms to enhance performance in an electronic nose," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 4, pp. 429-439, 1999.

[37] W. P. Carey, K.R. Beebe, and B.R. Kowalski, "Multicomponent analysis using an array of piezoelectric crystal sensors," *Anaytical* Chemistry, vol. 59, pp. 1529-1534, 1987.

[38] H. Ulmer, J. Mitrovics, G. Noetzel, U. Weimar, and W. Göpel, " Odours and flavours identified with hybrid modular sensor systems," *Sensors and Actuators B*, vol. 43, pp. 24-33, 1997.

[39] S. V. Patel, M.W. Jenkins, R.C. Hughes, W.G. Yelton, and A.J. Ricco, "Differentiation of chemical components in a binary solvent vapor mixture using carbon/polymer composite based chemiresistors," *Analytical Chemistry*, vol. 72, pp. 1532-1542, 2000.

[40] E.T. Zellers and M Han, "Effects of temperature and humidity on the performance of polymer coated surface acoustic wave vapor sensor arrays," *Analytical Chemistry*, vol. 68, pp. 2409-2418, 1996.

[41] Y. Kato, K. Yoshikawa, and M. Kitora, " Temperature dependent dynamic response enables the qualification and quantification of gases by a single sensor," *Sensors and Actuators B*, vol. 40, pp. 33-37, 1997.

[42] G. Faglia, F. Bicelli, G. Sberveglieri, P. Maffezoni, and P. Gubian, " Identification and quantification of methane and ethyl alcohol in an environment at variable humidity by an hybrid array," *Sensors and Actuators B*, vol. 44, pp. 517-520, 1997.

[43] M.C. Horrillo, J. Getino, J. Gutièrrez, L. Arès, J.I. Robla, C. Garcia, and I. Sayago, " Measurements of VOCs in soils through a tin oxide multisensor system," *Sensors and Actuators B*, vol. 43, pp. 193-199, 1999.

[44] J.W. Grate and M.H. Abraham, "Solubility interactions and the design of chemically selective sorbent coatings for chemical sensors and arrays," *Sensors and Actuators B*, vol. 3, pp. 85-111, 1991.

[45] S. J. Patrash and E.T. Zellars, " Characterization of polymeric surface acoustic wave sensor coatings and semiempirical models of sensor responses to organic vapors," *Analytical Chemistry*, vol. 65, pp. 2055-2066, 1993.

[46] J.W. Grate, M.H. Abaraham, and R.A. McGill, " Sorbent polymer materials for chemical sensors and arrays," In *Handbook of Biosensors and Electronic Noses* (Ed. E.K. Rogers), ch. 25, pp.593-612, CRC Press: Boca Raton, FL, 1997.

[47] E. Marieb, In *Human Anatomy and Physiology, 4th Edition*, ch. 16, pp. 539-541, Benjamin/Cummings Science Publishing: Menlo Park, CA, 1998.

[48] G.J. Tortora and S. R. Grabowski, In *Principles of Anatomy and Physiology, 8th Edition*, ch. 16, pp. 453-455, Harper Collins College Publishers: New York NY, 1996.

[49] L.R. Johnson, In *Essential Medical Phyiology, 2nd Edition*, ch.54, pp. 746-749, Lippincott-Raven: Philadelphia, PA, 1998.

[50] W. Ganong, In *Review of Medical Physiology, 19th Edition*, ch. 10, pp. 175-178, Appleton & Lange: Stamford, CT, 1999.

[51] R.R. Seeley, T.D. Stephens, and P. Tate, In *Anatomy and Physiology, 3rd Edition*, ch. 15, pp. 475-477, Mosby Year Book: St. Louis, MO, 1995.

[52] H. Breer, "Sense of smell: Signal recognition and transduction in olfactory receptor neurons," In *Handbook of Biosensors and Electronic Noses (Ed. E. Rogers)*, ch. 22, pp. 521-532, CRC Press: Boca Raton FL, 1997.

[53] F. Martini, K. Welch, and W. Ober, In *Fundamentals of Anatomy and Physiology, 4th Edition*, ch 17., Prentice Hall: Upper Saddle River, NJ, 1998.

[54] E.B. Keverne, " The vomeronasal organ," *Science*, vol. 286, pp. 716-720, 1999.

[55] K. Mori, H. Nagao, and Y. Yoshihara, " The olfactory bulb: Coding and Processing of odor molecule information", *Science*, vol. 286, pp. 711-715, 1999.

[56] P.E. Keller, "Physiologically inspired patterns recognition for electronic noses," *Proceedings of SPIE, Applications and Science of Computation Intelligence II*, vol. 3722, no. 13, pp. 144-153, 1999.

[57] D. Kohl, "Semiconductor and calorimetric sensor devices and arrays," In *Handbook of Biosensors and Electronic Noses* (Ed. E. Rogers), ch. 23, pp. 533-561, CRC Press, Boca Raton, FL, 1997.

[58] K.C. Persaud and P.Pelosi, "Sensor arrays using conducting polymers for an artificial nose," In *Sensors and Sensory Systems for an Electronic Nose (Ed. J.W. Gardner and P.N. Basrtlett)*, ch. 15, pp. 237-256, NATO ASI Series, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.

[59] J. Mitrovics, H. Ulmer, U. Weimar, and W. Göpel, "Modular sensor systems for gas sensing and odor monitoring: The MOSES concept," *Accounts of Chemical Research*, vol. 31, no. 5, pp. 307-315, 1998.

[60] T. Wu, "A piezoelectric biosensor as an olfactory receptor for odor detection: electronic nose," *Biosensors and Bioelectronics*, vol. 14, pp. 9-18, 1999.

[61] A. Mandelis and C. Christofides, *Physics, Chemistry and Technology of Solid State Gas Sensor Devices*, John Wiley and Sons: New York, NY, 1993.

[62] H.T. Nagle, R. Gutierrez-Osune, S.S. Schiffman, "The how and why of electronic noses," *IEEE Spectrum*, vol. 35, no. 9, pp. 22-34, 1998.

[63] I. Greenberg, "A nose for business," *Technology Review*, vol.10, no.4, pp.62-67, 1999.

[64] E. Rogers (editor), *Handbook of Biosensors and Electronic Noses*, CRC Press: Boca Raton, FL, 1997.

[65] W. King, "Piezoelectric sorption detector," *Analytical Chemistry*, vol. 36, no. 9, pp. 1735-1739, 1964.

[66] G. Z. Sauerbrey, "Use of vibrating quartz for thin film weighting and microweighing (in German)," *Zeitchrift für Physik*, vol. 155, pp. 206-222, 1959.

[67] A. D'Amico, C.D. Natele, E. Verona, "Acoustic Devices," In *Handbook of Biosensors and Electronic Noses (Ed. K. Rogers)*, Ch. 9, pp. 197-223, CRC Press: Boca Raton, FL, 1997.

[68] J.W. Grate, S.J. Martin, and R.M. White, "Acoustic wave microsensors, part I," *Analytical Chemistry*, vol. 65, no. 21, pp. 940A-948A, 1993.

[69] J.W. Grate, S.J. Martin, and R.M. White, "Acoustic wave microsensors, part II," *Analytical Chemistry*, vol. 65, no. 22, pp. 987A-996A, 1993.

[70] J.W. Grate, B.M. Wise, and M.H. Abraham, "Method for unknown vapor characterization and classification using a multivariate sorption detector. Initial derivation and modeling based on polymer coated acoustic wave sensor arrays and linear solvation energy relationships," *Analytical Chemistry*, vol. 71, no. 20, pp. 4544-4553, 1999.

[71] D. S. Ballantine, S.L. Rose, J.W.Grate, and H. Wohltjen, "Correlation of surface acoustic wave device coating responses with solubility properties and chemical structure using pattern recognition," *Analytical Chemistry*, vol. 58, no. 14, pp. 3058-3066, 1986.

[72] R. Lucklum, C. Behling, R.W. Cernosek, and S.J. Martin, "Determination of complex shear modulus with thickness shear mode resonators," *Journal of Physics D: Applied Physics*, vol. 30, pp. 346-356, 1997.

[73] K.W. Whitten, K.D. Gailey, and R.E. Davis, *General Chemistry with Qualitative Analysis, 4th Edition*, Saunders College Publishers: Fort Worth, TX, 1992.

[74] R.T. Morrison, and R.N. Boyd, *Organic Chemistry, 6th Edition*, Prentice Hall: Englewood Cliffs, NJ, 1992.

[75] R. Shinar, Personal communications, 1997-2000.

[76] J.D.N. Cheeke and Z. Wang, "Acoustic wave gas sensors," *Sensors and Actuators B*, vol. 59, pp. 146-153, 1999.

[77] M. Rodahl, F. Höök, and B. Kasemo, "QCM Operation in liquids: An explanation of measured variations in frequency and Q factor with liquid conductivity," *Analytical Chemistry*, vol. 68, no. 13, pp. 2219-2227, 1996.

[78] C.K. O'Sullivan and G.G. Guilbault, "Commercial quartz crystal microbalances – theory and applications," *Biosensors and Bioelectronics*, vol. 14, pp. 663-670, 1999.

[79] M.H. Abraham, "Scales of hydrogen-bonding: Their construction and application to physicochemical and biochemical processes," *Chemical Society Reviews*, vol. 22, pp. 73-83, 1993.

[80] H. Wohltjen, "Mechanism of operation and design considerations for surface acoustic wave device vapor sensors," *Sensors and Actuators*, vol. 5, pp. 307-325, 1984.

[81] J.T. Tou and R.C. Gonzales, *Pattern Recognition Principles*, Addison Wesley Publishing Company: Reading, MA, 1974.

[82] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press: San Diego, CA, 1990.

[83] R.J. Schalkoff, *Pattern Recognition: Statistical, Structural, and Neural Approaches*, John Wiley and Sons: New York, NY, 1991.

[84] L. Zadeh, "Fuzzy sets," *Information Control*, vol. 8, pp. 338-352, 1965.

[85] J-S.R. Jang, C.-T. Sun, and E. Mizutani, E., *Neuro-fuzzy and Soft Computing, A Computational Approach to Learning and Machine Intelligence*; Prentice Hall: Upper Saddle River, NJ, 1997.

[86] C.H. Chen, Ed. *Fuzzy Logic and Neural Network Handbook*, McGraw Hill: New York, 1996.

[87] C.-T. Lin, C.S.G. Lee, *Neural Fuzzy Systems, A Neuro-Fuzzy Synergism to Intelligent Systems*, Prentice Hall: Upper Saddle River, NJ, 1996.

[88] R. Lippmann, "An introduction to computing with neural nets," *IEEE Acoustics, Speech and Signal Processing Magazine*, vol. 4, pp. 4-22, 1987.

[89] D.R. Hush and B.G., Horne, "Progress in supervised neural networks," *IEEE Signal Processing Magazine*, vol. 10, no.1, pp.8-39, 1993.

[90] S. Haykin, *Neural Networks, A Comprehensive Foundation, 2nd Edition*, IEEE Press: Piscataway, NJ, 1999.

[91] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 1-47, 1998.

[92] B. Schölkopf, C.J.C. Burges, A.J. Smola (editors), In *Advance in Kernel Methods, Support Vector Learning*, MIT Press: Cambridge, MA, 1999.

[93] J.T. Tou and R.C. Gonzales, *Pattern Recognition Principles*, Addison Wesley Publishing Company: Reading, MA, 1974.

[94] R. O. Duda , D.G. Stork, and P.E. Hart, *Pattern Classification, 2nd Edition*, John Wiley and Sons: New York NY, 2000.

[95] V. Chandrasekaran and Z. Kiu, "Improving linear separability of classes via feature projection, " *Proceedings of the IEEE International Conference on Intelligent Processing Systems*, pp. 958-962, 1997.

[96] K. I. Diamantaras and M.G. Strintzis, "Neural nonlinear classifiers with synaptic weight commitment," *Proceedings of the 1997 IEEE International Symposium on Circuits and Systems*, pp. 653-656, 1997.

[97] K. I. Diamantaras and M.G. Strintzis, "Neural classifiers using one-time updating," *IEEE Transactions on Neural Networks*, vol. 9, no. 3, pp. 436-447, 1998.

[98] G.C. Osbourne and R.F. Martinez, "Empirically defined regions of influence for clustering analysis," *Pattern Recognition*, vol. 28, no. 11, pp. 1793-1806, 1995.

[99] G.C. Osbourn, J.W. Bartholomew, A.M. Bouchard, and R.F. Martinez, "Automated pattern recognition based on the visual empirical region of influence (VERI) method: A user's guide," *Technical report from Sandia National Laboratories at* http://www.sandia.gov/imrl/XVisionScience/Xusers.htm.

[100] G.C. Osbourn, J.W. Bartholomew, A.J. Ricco, and G.C. Frye, " Visual-empirical region of influence pattern recognition applied to chemical microsensor array selection and chemical analysis," *Accounts of Chemical Research*, vol. 31, pp. 297-305, 1998.

[101] S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, pp. 783-789, 1999.

[102] M. Sellathurai and S. Haykin, "The separability theory of hyperbolic tangent kernels and support vector machines for patterns classification," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP99)*, vol. 2, pp. 1021-1024, 1999.

[103] J. Proakis, *Digital Comunications*, McGraw Hill: New York, NY, 1989.

[104] R. Gonzales and R. Woods, *Image Processing*, pp. 171-180, Addison Wesley, Reading, MA, 1992.

[105] G. Casella, G. R.L. Berger, *Statistical Inference*, pp. 45-54, Duxbury Press, Belmont, CA, 1990.

[106] D.F. Specht, "A general regression neural network", *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568-576, 1991.

[107] P.D. Wasserman, *Advanced Methods in Neural Computing*, pp. 155-173, Van Nostrand Reinhold, New York, 1993.

[108] D.F. Specht, "Probabilistic neural networks", *Neural Networks*, vol. 3, pp. 109-118, 1990.

[109] A.J. Ricco, R.C. Crooks, and G.C. Osbourn, "Surface acoustic wave chemical sensor arrays: New chemically sensitive interfaces combined with novel cluster analysis to detect volatile organic compounds and mixtures," *Accounts of Chemical Research*, vol. 31, no. 5, pp. 289-296, 1998.

[110] J. Park and E.T. Zellers, "Determining the minimum number of sensors required for multiple vapor recognition with arrays of polymer coated SAW sensors," *Proceedings of the Electrochemical Society*, vol. 99-23, pp. 132-137, 1999.

[111] J. Park, W.A. Groves, and E.T. Zellers, "Vapor recognition with small arrays of polymer coated microsensors," *Analytical Chemistry*, vol. 71, no. 17, pp. 3877-3886, 1999.

[112] G.C. Osbourn, R.F. Martinez, J.W. Bartholomew, W.G. Yelton, and A.J. Ricco, "Optimizing chemical sensor array," *Proceedings of the Electrochemical Society*, vol. 99-23, pp. 127-131, 1999.

[113] W. P. Carey, K.R. Beebe, B.R. Kowalski, D.L. Illman, and T. Hirschfeld, "Selection of adsorbates for chemical sensor arrays by pattern recognition," *Analytical Chemistry*, vol. 58, no. 1, pp. 149-153, 1986.

[114] F. Avila, D.E. Myers, and C. Palmer, "Correspondence analysis and adsorbate selection for chemical sensor arrays," *Journal of Chemometrics*, vol. 5, pp. 455-465, 1991.

[115] B.G. Kermani, S.S. Schiffman, and H.T. Nagle," A novel method for reducing the dimensionality in a sensor array," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, no. 3, pp. 728-740, 1998.

[116] M. Marth, D. Maier, U. Stahl, M. Rapp, T. Wessa, and J. Honerkamp, "Optimization of surface acoustic wave sensor arrays and application to high performance liquid chromatography," *Sensors and Actuators B*, vol. 61, pp. 191-198, 1999.

[117] D.M. Wilson, K. Dunman, T. Roppel, R. Kalim, "Rank extraction in tin oxide sensor arrays," *Sensors and Actuators B*, vol. 62, pp. 199-210, 2000.

[118] T.M. Mitchell, *Machine Learning*, WCB/McGraw Hill: Boston, MA. 1997.

[119] J.R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.

[120] D.W. Aha and R.L. Blanket, "A comparative evaluation of sequential feature selection algorithms," *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics*, pp. 1-7, 1995.

[121] B.V. Dasarathy, *Nearest Neighborhood (NN) Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press: Los Alamitos, CA, 1990.

[122] H. Almuallim, T.G. Dietterich, "Learning Boolean concepts in the presence of many irrelevant features," *Artificial Intelligence*, vol. 69, no. 1-2, pp. 279-306, 1994.

[123] K. Kira and K, L.A. Rendell, "Feature selection problem: traditional methods and a new algorithm," *Proceedings 10th International Conference on Machine Learning (AAAI92)*, pp. 129-134, 1992.

[124] D. Koller and M. Sahami, "Toward optimal feature selection," P*roceedings of the 13th International Conference on Machine Learning (ICML-96)*, pp. 284-292, Morgan Kaufmann: San Francisco, CA. 1996.

[125] K. Etemad and R. Chellappa, " Dimensionality reduction of multi-scale feature spaces using a separability criterion," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP95)*, vol. 4, pp. 2547-2550, 1995.

[126] K. Etemad and R. Chellappa, " Separability based multiscale basis selection and feature extraction for signal and image classification," *IEEE Transactions on Image Processing*, vol. 7, no. 10, pp. 1453-1465, 1998.

[127] M.V. Wickerhauser. *Adaptive Wavelet Analysis from Theory to Software*, IEEE Press: Piscataway, NJ, 1994.

[128] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithms," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 2, pp. 44-49, 1998.

[129] R. Kohavi and G.H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273-324, 1997.

[130] H. Liu and R. Setiono, "Incremental feature selection," *Applied Intelligence*, vol. 9, no. 3, pp. 217-230, 1998.

[131] R. Setiono and H. Liu, "Neural network feature selector," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 654-662, 1997.

[132] J.R. Quinlan, *C 4.5: Programs for Machine Learning*, Morgan Kaufmann: San Mateo, CA, 1993.

[133] M. Seo, *Automatic Ultrasonic Signal Classification Scheme*, MS Thesis, Iowa State University, 1997.

[134] R. Kohavi and G.H. John, In *Feature Extraction, Construction and Selection: A Data Mining Perspective, (Ed. L. Huan and M. Hiroshi)*, Kluwer Academic Publishers, Norwell MA, 1998.

[135] P. Jantke, "Types of incremental learning", *AAAI Symposium on Training Issues in Incremental Learning*, 1993.

[136] B. Zhang, "An incremental learning algorithm that optimizes network size and sample size in one trial," *Proceedings of the IEEE International Conference on Neural Networks (ICNN94)*, pp. 215-220, 1994.

[137] E.H. Wang and A. Kuh, "A smart algorithm for incremental learning," *Proceedings of International Joint Conference on Neural Networks, (IJCNN92)* vol. 3, pp.121-126, 1992.

[138] F. S. Osorio and B. Amy, "INSS: A hybrid system for constructive machine learning," *Neurocomputing*, vol. 28, pp. 191-205, 1999.

[139] M.T. Vo, "Incremental learning using the time delay neural network," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP94)*, vol. 2, pp. 629-632, 1994.

[140] T. Hoya and A. G. Constantidines, "A Heuristic pattern correction scheme for GRNNs and its application to speech recognition," *Proceedings of the1998 IEEE Signal Processing Society Workshop*, pp.351-359, 1998.

[141] C. H. Higgins and R.M. Goodman, "Incremental learning with rule-based neural networks," *Proceedings of International Joint Conference on Neural Networks (IJCNN91)*, vol. 1, pp. 875-880, 1991.

[142] K. Yamauchi and N. Ishii, "An incremental learning method with recalling interfered patterns," *Proceedings of the IEEE International Conference on Neural Networks (ICNN95)* vol. 6, pp. 3159-3164, 1995.

[143] K. Yamauchi, N. Yamaguchi, and N. Ishii, "Incremental learning methods with retrieving of interfered patterns," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp.1351-1365, 1999.

[144] N. Ramani, "Incremental learning using hidden layer activations – tests on fish identification data," *Proceedings of International Joint Conference on Neural Networks (IJCNN92)*, vol.1, pp. 640-645, 1992.

[145] G. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen, "Fuzzy ARTMAP: An adaptive resonance architecture for incremental learning of analog maps," *Proceedings of International Joint Conference on Neural Networks (IJCNN92)*, vol. 3, pp. 309-314, 1992.

[146] G. A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 698-713, 1992.

[147] H. Chen and V Soo, "Design of adaptive and incremental feed-forward neural networks," *Proceedings of IEEE International Conference on Neural Networks (ICNN93)*, pp. 479-484, 1993.

[148] L. Fu, H. Hsu, and J.C. Principe, "A knowledge-based approach to supervised incremental learning," *Proceedings of IEEE International Conference on Neural Networks and World Congress on Computational Intelligence*, vol. 3, pp. 1793-1798, 1994.

[149] L. Fu, "Incremental knowledge acquisition in supervised learning networks," *IEEE Transactions on Systems, Man, and Cybernatics – Part A: Systems and Humans"* vol. 26, no. 6, pp. 801-809, 1996.

[150] L. Fu, H. H. Hsu, and J.C. Principe, "Incremental backpropagation learning networks," *IEEE Transactions on Neural Networks*, vol. 7, no. 3, pp. 757-762, 1996.

[151] G. Tontini, "Robust learning and identification of patterns in statistical process control charts using a hybrid RBF fuzzy artmap neural network," *Proceedings of International Joint Conference on Neural Networks (IJCNN98)*, vol. 3, pp. 1694-1699, 1998.

[152] L. Bruzzone, D. F. Prieto, and R. Cossu, "An incremental learning classifier for remote-sensing images," *Proceedings of IEEE International Geoscience and Remote Sensing Symposium (IGARSS99)*, vol. 5, pp. 2483-2485, 1999.

[153] J.F. Hebert, M. Parizeau, and N. Ghazzali, "Cursive character detection using incremental learning," *Proceedings of the 5th International Conference on Document Analysis and Recognition(ICDAR99)*, pp. 390-393, 1999.

[154] A. Vailaya and A. Jain, " Incremental learning for Bayesian classification of images," *Proceedings of International Conference on Image Processing (ICIP99)*, vol. 2, pp. 585-589, 1999.

[155] S. Vijayakumar and H. Ogawa, "RKHS-based functional analysis for exact incremental learning," *Neurocomputing*, vol. 29, pp. 85-113, 1999.

[156] R. Schapire, "Strength of weak learning," *Machine Learning*, vol. 5, pp. 197-227, 1990.

[157] N. Littlestone and M. Warmuth, "Weighted majority algorithm, "*Information and Computation*, vol. 108, pp. 212-261, 1994.

[158] Y. Freund, "Boosting a weak learning algorithm by majority", *Information and Computation*, vol. 2, pp. 121, 1996.

[159] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," *Proc. of the 13th International Conference on Machine Learning*, pp.148-156, 1996.

[160] Y. Freund and R. Schapire, "A decision theoretic generalization of on-line learning and an application to boosting," *Computer and System Sciences,* vol. 57, no. 1, pp. 119-139, 1997.

[161] R. Schapire, Y. Freund, P. Bartlett, and W.S. Lee, " Boosting the margins: a new explanation for the effectiveness of voting methods," *The Annals of Statistics*, vol. 26, no. 5, pp. 1651-1686, 1998.

[162] R. Schapire and Y. Singer, " Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 39, no. 2/3, pp. 135-168, 1999.

[163] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.

[164] J.R. Quinlan, "Bagging, boosting and C4.5, " *Proceedings of the National Conference on Artificial Intelligence*, vol. 1, pp. 725-730, 1996.

[165] T.G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization," *Machine Learning*, vol. 40, no. 2, pp. 1-19, 2000.

[166] D.H. Wolpert, "Stacked Generalization," *Neural Networks,* vol.5, no. 2, pp.241-259, 1992.

[167] J. Kittler, M. Hatef, R.P. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no.3, pp. 226-239, 1998.

[168] S. Rangarajan, P. Jalote, and S. Tripathi, "Capacity of voting systems," *IEEE Transactions on Software Engineering*, vol. 19, no. 7, pp. 698-706, 1993.

[169] C. Ji and S. Ma, "Combination of weak classifiers," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 32-42, 1997.

[170] K.M. Ali, C. Brunk, and M.J. Pazzani, "On learning multiple descriptions of a concept," *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, pp. 476-483, 1994.

[171] K.M. Ali and M.J. Pazzani, "Error reduction through learning multiple descriptions," *Machine Learning*, vol. 24, no. 3, pp. 173-202, 1996.

[172] C. Ji and S. Ma, "Performance and efficiency: recent advances in supervised learning," *IEEE Proceedings*, vol.87, no. 9, pp. 1519-1535, 1999.

[173] T.G. Dietterich, "Machine learning research," *AI Magazine*, vol. 18, no. 4, pp. 97-136, 1997.

[174] M. J. Kearns and U.V. Vazirani, In *An introduction to computational learning theory*, MIT Press: Cambridge, MA, 1994.

[175] V. Honavar, "Computational learning theory," In *Lecture Notes for ComS 672: Computational Models of Learning*, available at class web site http://www.cs.iastate.edu/~honavar/Courses/cs672/weekly.html

[176] C.L. Blake and C.J. Merz, UCI Repository of machine learning databases at http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science.

[177] R. DeNale and C. A. Lebowitz, "Ultrasonics as an alternative to radiography for submarine hull weld inspection," *Ship Materials Engineering Department Research and Development Report, US Navy David Taylor Research Center*", February 1990.

[178] P. Ramuhalli, J. Kim, P. Cui, R. Polikar, L. Udpa, S. S. Udpa, T. Trapp, M. Novack, H. Castner , R. Spencer, R. Kok, "Knowledge-based ultrasonic weld inspection system," 27th Annual Review of Progress in Quantitative Nondestructive Evaluation, (to appear), 2000.

[179] P. Ramuhalli, *Automatic signal classification systems using fuzzy ARTMAP networks*, M.S. thesis, Iowa State University, 1998.